Wireless multi-carrier communication system design and implementation using a custom hardware and software FPGA platform

by

Marcus Robert Perrett

A thesis submitted for the degree of Doctor of Engineering (EngD)

Communications and Informations Systems Research Group

Department of Electrical and Electronic Engineering

University College London

July 29, 2012

Statement of Originality

I, Marcus Robert Perrett confirms that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the thesis.

Signed: ____________________ Date: ____________________

Abstract

Field Programmable Gate Array (FPGA) devices and high-level hardware development languages represent a new and exciting addition to traditional research tools, where simulation models can be evaluated by the direct implementation of complex algorithms and processes. Signal processing functions that are based on well known and standardised mathematical operations, such as Fast Fourier Transforms (FFTs), are well suited for FPGA implementation. At UCL, research is on-going on the design, modelling and simulation of Frequency Division Multiplexing (FDM) techniques such as Spectrally Efficient Frequency Division Multiplexing (SEFDM) which, for a given data rate, require less bandwidth relative to equivalent Orthogonal Frequency Division Multiplexing (OFDM). SEFDM is based around standard mathematical functions and is an ideal candidate for FPGA implementation. The aim of the research and engineering work reported in this thesis is to design and implement a system that generates SEFDM signals for the purposes of testing and verification, in real communication environments. The aim is to use FPGA hardware and Digital to Analogue Converters (DACs) to generate such signals and allow reconfigurability using standard interfaces and user friendly software. The thesis details the conceptualisation, design and build of an FPGA-based wireless signal generation platform. The characterisation applied to the system, using the FPGA to drive stimulus signals is reported and the thesis will include details of the FPGA encapsulation of the minimum protocol elements required for communication (of control signals) over Ethernet. Detailed testing of the hardware is reported, together with a newly designed *in the loop* testing methodology. Verified test results are also reported with full details of time and frequency results as well as full FPGA design assessment. Altogether, the thesis describes the engineering design, construction and testing of a new FPGA hardware and software system for use in communication test scenarios, controlled over Ethernet.

Acknowledgements

I would like to thank my supervisor, Professor Izzat Darwazeh, for his support, guidance and council over the period of my EngD. In particular, I am grateful for his suggestion of the EngD to me and the positive impact it has had on my career. I have also greatly benefitted from his extensive experience in telecommunications, industry and life in general. I am also grateful for the funding he has secured for my studies throughout my EngD.

My gratitude is extended to my fellow students that have provided support, collaboration and fruitful discussions over the time of my EngD. I thank Ryan Grammenos for his help and guidance in devising a verification strategy together and for his continuation of the hardware implementation of SEFDM. I would like to thank Safa Isam (now Dr Isam) for helping target her work for hardware implementation and patiently explaining the subtleties to me. I also would like to thank Paul Whatmough for his effort on the analysis and optimisation of my work and taking it further to lead to our collaboration on a 2012 IEEE journal paper addressing Application-Specific Integrated Circuit (ASIC) implementation. Finally, I would like to thank Paul Mckenna for his management and co-ordination of the EngD. I extend that thanks to the EE administration team who have made my time at UCL entertaining and exciting.

I would like to thank my friends and family who have demonstrated extraordinary patience and understanding during my thesis writing. A special thank you to my nephew Finlay who, in years to come, will understand why his uncle missed important family gatherings. A special thanks goes to my wife to be Jenna for putting up with lost weekends and evenings and for proof reading the thesis. She has given me the motivation and inspiration to complete the EngD, while maintaining the level of quality such an endeavour deserves. Finally, I express my gratitude to the EPSRC for funding my EngD. I believe this type of funded education in collaboration with industry is hugely beneficial to all parties and I feel its a shame that there will be few future telecommunications EngD students.

Contents

Statement of Originality **1**

Abstract **2**

Acknowledgements **3**

Contents **4**

List of Figures **8**

List of Tables **14**

Abbreviations **16**

1 Introduction **21**

1.1 Motivation and Research Aims 23

1.2 Thesis Structure 25

1.3 Main Contributions 26

1.4 List of Publications 28

2 The Design and Build of an FPGA-Based Signal Generator **30**

2.1 The State of the Art 32

2.2 Specification 34

2.3 Operation 34

2.4 Component Selection 36

2.4.1 Ethernet Hardware Interface 37

2.4.2 Programmable Logic Device 39

2.4.3 Digital to Analogue Conversion 42

2.4.4 Memory 47

2.4.5 Component Summary 48

2.4.6 Power 49

2.4.7 Design Highlights 52

2.5 Schematic Design 54

2.6 PCB Design 56

2.7 Build 58

2.7.1 Enclosure 62

2.8 System Bring-up and Test 63

2.8.1 Power Supply Testing 64

2.8.2 FPGA Test Using Shell Code 67

2.8.3 Memory Testing 69

2.8.4 Ethernet Testing 70

2.8.5 DAC Testing 70

2.9 Conclusions 72

3 Characterisation of an FPGA-Based Signal Generator 74

3.1 Transient Measurements 77

3.2 Frequency Response Measurements 80

3.3 Noise and Distortion 82

3.4 Amplitude Linearity 85

3.5 Conclusions 87

4 A Simple Ethernet Stack Implementation in VHDL 88

4.1 Communication Over Ethernet 91

4.2 Address Resolution Protocol 93

4.2.1 Summary 95

4.3 FPGA Implementation 95

4.3.1 Stack Architecture 96

4.4 Ethernet Stack Evaluation 101

4.4.1 Functional Evaluation 101
4.4.2 Performance Evaluation 103
4.4.3 Critical Analysis 107
4.5 Conclusions 108

5 Hardware Implementation of SEFDM 110

5.1 Non-Orthogonal Modulation Techniques 114
5.1.1 Faster-Than-Nyquist Method 114
5.1.2 Direct Synthesis as a Bank of Modulators 116
5.1.3 Spectrally Efficient Frequency Division Multiplexing 119
5.1.4 Computational Complexity Comparison 120
5.2 SEFDM Implementation Targeted for FPGA 122
5.2.1 Modulo Arithmetic Implementation 123
5.2.2 Division versus Index Implementation Comparison 126
5.2.3 Phase Shift of IFFT Output Vectors 128
5.2.4 The Effect of Sub-carrier Compression on FPGA Resources . . 128
5.2.5 Signal Scaling 130
5.3 Functional Design Simulation 132
5.4 Hardware Design Results 135
5.5 Implementation Results 136
5.6 Critical Evaluation 139
5.7 Conclusions 140

6 Verification of a Hardware-Based SEFDM Transmitter 142

6.1 Verification Methodology 144
6.1.1 General Description 145
6.2 Transmitter Verification 149
6.3 Sampling Rate Discussion 153
6.4 Detector 155
6.5 Results 157
6.6 Conclusions 159

7 Conclusions **161**

7.1 Thesis Achievements and Comments 163

7.2 Ongoing Related Work at UCL 165

7.3 Proposals for Future Work 165

Appendix A M-Sequence Generator Paper **167**

Appendix B PCB Schematic Files **172**

Appendix C PCB Gerber Files **182**

Appendix D SEFDM Generator GUI Datasheet **195**

Appendix E Source code **204**

E.1 VHDL Code 205

E.2 Gerber Code 206

E.3 C++ Code 207

Appendix F Cordic Details **208**

References **210**

List of Figures

2.1 Conceptual system block diagram 35

2.2 Block Diagram of Ethernet Physical and MAC 37

2.3 Circuit Diagram of a Gigabit Ethernet RJ45 connector showing the voltage transformer and differential data carrying pairs 38

2.4 Figure showing the Xilinx Virtex 5 family breakdown (from vendor datasheet) 40

2.5 Block diagram of standard clock interface requirements from pin to internal clock tree (parallel lines are differential signals) 41

2.6 Figure showing a typical connection between an FPGA and an external flash memory (from vendor datasheet) 42

2.7 Figure showing SNR for increasing bit-resolution (green) with a corresponding reduction in available bandwidth (blue) 43

2.8 Figure showing the basic operation of the AD9779 (from vendor datasheet) 43

2.9 Figure showing the IQ modulator external connections and internal function (from vendor datasheet) 44

2.10 Figure showing a typical connection between the DAC and the IQ modulators (from vendor datasheet) 44

2.11 Block diagram showing the DAC output connection to the IQ modulator and buffer amplifiers 45

2.12 Figure showing the 2's complement input of the DAC and the corresponding output voltage 46

2.13 Figure showing the output of the Xilinx xpower tools for power estimation of the FPGA 49

2.14 Figure showing the ground plane isolation and potential connection points: on the regulator board (A) and after the AC-DC converters (B). Also shown are correct (green) and incorrect (red) Printed Circuit Board (PCB) track routes 51
2.15 Figure displaying the similar decoupling requirements for the IQ Modulator (left) and the Ethernet PHYCeiver (right) taken from the respective datasheets 52
2.16 Figure showing the reset strategy employed, using the FPGA to drive logic high or low as appropriate 53
2.17 Figure showing the schematic for DACs 1 and 2, together with IQ modulators 1 and 2 55
2.18 Figure illustrating the PCB design with all layers visible 58
2.19 Figure displaying the unpopulated power PCB top (left) and bottom (right) 59
2.20 Figure showing the top (left) and bottom (right) of the rework PCB. The top has the component soldered to it and the bottom is soldered to the main PCB 60
2.21 Figure illustrating the Ethernet device fitted to the pin translation PCB due to an incorrect main PCB footprint 60
2.22 Figure illustrating main PCB top layer (top) and bottom layer (bottom) populated with components after manufacture 61
2.23 Figure showing the power board populated with components after manufacture 62
2.24 Figure displaying the main and power PCB with mounting plates fitted (right) and the enclosure before the PCBs are inserted (Left) 63
2.25 Figure illustrating each cable from the main PCB (top) and the front panel of the enclosure with Sub-Minature "A" (SMA) connectors (bottom) . . 64
2.26 Figure showing the final configuration of the two PCBs and the enclosure 65
2.27 Figure showing the modification to the digital 1.0v power supply to rectify the incorrect layout. 67

2.28 Figure showing the FPGA Joint Test Action Group (JTAG) configuration as a daisy-chain of devices to a common header connection (taken from vendor datasheet) 68
2.29 Figure illustrating the AD9779 pinout (right) and the erroneous ground pin (left) (taken from vendor datasheet) 71
2.30 Figure to illustrate the modification to the DAC CGND and AGND pins to correct erroneous pin layout 72

3.1 Characterisation methodology and apparatus configuration 76
3.2 DAC 1 (top) to 4 (bottom) transient response of each I and Q output . . 79
3.3 DAC 1 (top) to 4 (bottom) frequency response of each I and Q output . 81
3.4 DAC 1 (top) to 4 (bottom) linearity response of each I and Q output . . 86

4.1 OSI networking model summary, noting direction data flows depending on receive or transmit operations 91
4.2 Ethernet, IP/ARP and TCP/UDP protocol byte-level configuration . . . 92
4.3 Details of Address Resolution Protocol (ARP) packet transactions (left) and byte-level description of ARP request and reply (right) 94
4.4 Logic Block Diagram of the Ethernet Stack operation between the Media Access Controller (MAC) and the user top level 96
4.5 State diagram of the MAC-to-user (RX) control logic 98
4.6 State diagram of the user-to-MAC (TX) control logic 99
4.7 Functional verification of a successful ARP transaction between a Host and Target 102
4.8 Functional verification of a packet not identified as an ARP being outputted to the user 104

5.1 SEFDM based transmitter 119
5.2 SEFDM (Type 1 and Type 2) Big $\mathcal{O}$ complexity analysis against the DDS method. OFDM based on a single IFFT is also included for comparison. 121
5.3 SEFDM (Type 2 only) Big $\mathcal{O}$ complexity analysis compared with DDS. Various sub-carrier granularity values are shown 121

5.4 Block Diagram of SEFDM transmitter operation block diagram 123
5.5 Logic topology diagram of an SEFDM transmitter 124
5.6 Logic diagram of Modulo divider circuit 125
5.7 Logic diagram of Modulo Index circuit 126
5.8 Chart showing the resource utilisation increase with respect to cMax . . 129
5.9 Figure to identify where scaling is required due to bit-growth in the SEFDM transmitter 131
5.10 Overview of the method employed to enable Matlab processing of simulation generated results 133
5.11 Power spectral plots for SEFDM with OFDM and Fast Orthogonal Frequency Division Multiplexing (FOFDM) equivalent values of α 134
5.12 Plot showing actual FPGA resource utilisation for varying bandwidth reduction and sub-carrier values of SEFDM 135
5.13 Plot showing actual and estimated resource utilisation for varying bandwidth reduction and sub-carrier values of SEFDM and Direct Digital Synthesis (DDS) 136
5.14 GUI settings used for the illustration of SEFDM generation 137
5.15 GUI setting and resulting spectrum for Alpha = 1 137
5.16 GUI setting and resulting spectrum for Alpha = 0.8 138
5.17 GUI setting and resulting spectrum for Alpha = 0.67 138
5.18 GUI setting and resulting spectrum for Alpha = 0.5 138
5.19 GUI setting and resulting spectrum for Alpha = 0.3 138

6.1 Verification methodology block diagram 145
6.2 Comparison of experimental data compared to analytical and FPGA architectural models for sequence ID 1 150
6.3 Cumulative absolute magnitude error for experimental and architectural signals with different bit scaling against analytical signals 151
6.4 Error performance with different internal and external data amplitudes. Data representation in numerical decimal internal to the FPGA 152

6.5 Error performance of new bit-scaling applied to the experimental data compared to architectural model estimate 153
6.6 Diagram showing the Sphere Decoding (SD) process 157
6.7 Detector results for data sequences in Table 6.1, quantified using analytical tool execution time 158
6.8 Detector results for data sequences in Table 6.1, quantified using minimum radius 159

B.1 Schematic page 1 - Top level component connections 173
B.2 Schematic page 2 - DAC 1/2 and IQ modulator 1/2 174
B.3 Schematic page 3 - DAC 3/4 and IQ modulator 3/4 175
B.4 Schematic page 4 - DAC power supplies and decoupling 176
B.5 Schematic page 5 - Ethernet PHYceiver 177
B.6 Schematic page 6 - FPGA I/O connectivity 178
B.7 Schematic page 7 - FPGA flash and JTAG interface 179
B.8 Schematic page 8 - DDR RAM 3/4 180
B.9 Schematic page 9 - DDR Ram 1/2 181

C.1 All 24-PCB layers together 182
C.2 Top (signal) layer 183
C.3 Signal layer 1 183
C.4 Signal layer 2 184
C.5 Signal layer 3 184
C.6 Signal layer 4 185
C.7 Signal layer 5 185
C.8 Signal layer 6 186
C.9 Signal layer 7 186
C.10 Signal layer 8 187
C.11 Signal layer 9 187
C.12 Signal layer 10 188
C.13 Signal layer 11 188
C.14 Signal layer 12 189

C.15 Signal layer 13 189
C.16 Power layer 1 190
C.17 Power layer 2 190
C.18 Power layer 3 191
C.19 Power layer 4 191
C.20 Power layer 5 192
C.21 Power layer 6 192
C.22 Power layer 7 193
C.23 Power layer 8 193
C.24 Bottom (signal) layer 194

D.1 Extended Ethernet diagram introducing the Stack and Top Level interfaces 196
D.2 Control GUI default startup screen 197
D.3 Alpha control area showing available sub-carrier compression fractions . . 198
D.4 Zero padding area showing available zero padding values. Padding is applied to both ends of the spectrum 199
D.5 Modulation control area showing available modulation schemes available 199
D.6 Debug area showing possible built-in signals that can be generated for characterisation purposes 200
D.7 Internal clock control area showing available clock rates the internal FPGA logic can run 201
D.8 External clock control area showing available clock rates the FPGA can supply data to the DAC 201
D.9 Data amplitude area showing real and imaginary amplitude adjustment . 202
D.10 Acquisition control area showing data sequence controlled by a counter (left) or an input integer value (right) 202
D.11 Example use case of real-only data with positive-only amplitude 203
D.12 Example use case of real-only data with positive and negative amplitude 203
D.13 Example use case of real and imaginary data combined with positive and negative amplitude 203

List of Tables

1.1 Table showing FPGA stimulus to perform characterisation measurements 26

2.1 Table comparing programmable logic device manufacturers available in 2007 33

2.2 Interconnect standards available for data transfer between a PHYCeiver and MAC 39

2.3 Table detailing the component selection for the major operations of the system 49

2.4 Table detailing the total power supply together with the target device and domain 50

2.5 Table detailing the initial power supply results per voltage under required current load 65

2.6 Table detailing the pin mismatching between the original 1.0v power device (LT1076) and the replacement part (MIC49300) 66

2.7 Table detailing the 1.0 power supply results after remedial modification of the PCB and the change of device (under current load) 66

3.1 DAC 1 Transient Results 78

3.2 DAC Frequency Response Results 82

3.3 SNR Results 84

3.4 SINAD Results 84

3.5 ENOB Results 84

3.6 Error for each DAC 87

4.1 Table detailing the resource breakdown of the stack 104

4.2 Table detailing the estimated resource utilisation from vendor tools . . . 105
4.3 Table comparing the features of this work compared to a fully featured solution 105
4.4 Table comparing the resource requirements of this work compared to a fully featured solution 106
4.5 Table detailing the timing performance estimation from vendor tools . . 106

5.1 Divisor versus index method for modulo matrix augmentation 127

6.1 Input data sequence selected 146
6.2 Highest performing variable settings used for the in the loop verification of SEFDM 148

D.1 Possible clock divider values and resulting clock frequency 201

E.1 VHDL Code heirarchy and Description 205
E.2 Gerber code description 206
E.3 C++ Code heirarchy and Description 207

Abbreviations

8-QAM 8-Quadrature Amplitude Modulation

16-QAM 16-Quadrature Amplitude Modulation

ADC Analogue To Digital Converter

ARP Address Resolution Protocol

ASIC Application-Specific Integrated Circuit

AWG Arbitrary Waveform Generator

BER Bit Error Rate

BOM Bill Of Materials

BPSK Binary Phase Shift Keying

CORDIC COordinate Rotation DIgital Computer

CPLD Complex Programmable Logic Device

CPU Central Processing Unit

CRC Cyclic Redundancy Check

CSMA/CD Carrier Sense Multiple Access / Collision Detection

DAC Digital to Analogue Converter

DDR Double Data Rate

DDS	Direct Digital Synthesis
DFT	Discrete Fourier Transform
DMM	Digital Multi Meter
DNL	Differential Non-Linearity
DRAM	Dynamic Random Access Memory
DSP	Digital Signal Processing
EMI	Electro-Magnetic Interference
ENOB	Effective Number Of Bits
FDM	Frequency Division Multiplexing
FF-LUT	Flip-Flop - Look Up Table
FFT	Fast Fourier Transform
FOFDM	Fast Orthogonal Frequency Division Multiplexing
FPGA	Field Programmable Gate Array
FTN	Faster-than-Nyquist
Gb	Gigabit
GbE	Gigabit Ethernet
GMII	Gigabit Media Independent Interface
GPU	Graphics Processing Unit
GUI	Graphical User Interface
HC-MCM	High-Compaction Multi-Carrier Modulation
IC	Integrated Circuit
ICI	Inter Carrier Interference

IDFT	Inverse Discrete Fourier Transform
IFFT	Inverse Fast Fourier Transform
INL	Integrative Non-Linearity
IOTA	Isotropic Orthogonal Transform Algorithm
IP	Internet Protocol
ISI	Inter Symbol Interference
JTAG	Joint Test Action Group
LED	Light Emitting Diode
LFSR	Linear Feedback Shift Register
LUT	Look-Up Table
LSB	Least Significant Bit
LTE	Long Term Evolution
LVDS	Low Voltage Differential Signalling
LVCMOS	Low Voltage Complimentary Metal Oxide Semiconductor
LVTTL	Low Voltage Transistor-Transistor Logic
MAC	Media Access Controller
MIG	Memory Interface Generator
MIMO	Multiple In Multiple Out
ML	Maximum Likelyhood
MMSE	Minimum Mean Square Error
NIC	Network Interface Card
OFDM	Orthogonal Frequency Division Multiplexing

OQAM Offset Quadrature Amplitude Modulation

OSI Open Systems Interconnect

OvFDM Overlapping Orthogonal Frequency Division Multiplexing

PAPR Peak to Average Power Ratio

PC Personal Computer

PCB Printed Circuit Board

PCS Physical Coding Sub-Layer

PLD Programmable Logic Device

PMA Physical Medium Attachment

PMD Physical Medium Dependant

PRBS Pseudo Random Binary Sequence

PROM Programmable Read Only Memory

QAM Quadrature Amplitude Modulation

QPSK Quadrature Phase Shift Keying

RAM Random Access Memory

RGMII Reduced Gigabit Media Independent Interface

RVD Real Value Decomposition

SD Sphere Decoder

SE Schnorr-Euchner

SEFDM Spectrally Efficient Frequency Division Multiplexing

SFDR Spurious Free Dynamic Range

SGMII Serial Gigabit Media Independent Interface

SINAD SNR Including Noise And Distortion

SMA Sub-Minature "A"

SMC Sub-Minature "C"

SNR Signal to Noise Ratio

SPI Standard Peripheral Interconnect

SQNR Signal-to-Quantisation Noise Ratio

SRAM Static Random Access Memory

TCP Transport Control Protocol

THD Total Harmonic Distortion

ToE TCP offload Engine

TSVD Truncated Singular Value Decomposition

UCL University College London

UDP User Datagram Protocol

USB Universal Serial Bus

VHDL Very high speed integrated circuit Hardware Description Language

ZF Zero Forcing

Chapter 1

Introduction

Field Programmable Gate Array (FPGA) technology has its roots originating from the beginnings of digital electronics. The inception of the transistor enabled two distinct features; that of amplification and that of a gate or switch. The latter capability diverged into microprocessor development for general purpose computing and general purpose logic devices containing logical expressions and was termed *Glue Logic*. Widespread use, particularly in industrial control, led to the development of the Programmable Logic Device (PLD), a device containing a subset of generalised expressions that could be configured externally via a graphical interface. This type of configuration was defined as *ladder logic*. In computer and video processing platforms, similar devices, commonly known as Programmable Read Only Memory (PROM), could be configured using a text-only language called *ABEL*. Vendors such as Xilinx, Altera and Lattice observed the market requirement (initially for military applications) for a more generalised and large capacity programmable logic device. The ease of programming found with PLDs and higher speed and density of PROMs were combined, resulting in the Complex Programmable Logic Device (CPLD). In parallel, programming languages such as Very high speed integrated circuit Hardware Description Language (VHDL) and Verilog were developed for high-level code abstraction. The syntax of such languages is very similar to C++, but a fundamental difference is that logic coding is based on parallel events and not hierarchical functions.

Internally, a CPLD device is a matrix of logical operations which overlap; connections between them can be *made* or *broken* via the programming language, as a logical expression is constructed. FPGA Vendors made efforts to produce faster speed and higher resource density devices by utilising improved lithography processes. Such vendors do not maintain the capability to produce devices themselves (so called fabless) and instead utilised advanced processes offered by fabrication companies such as TMSC. However, a limiting factor in the growth of CPLD devices was that, as size and density increase so does the requirement for internal logic interconnections. Programmable logic devices do not complete a single task, but rather many tasks in parallel. Therefore, signal routing between internal logic blocks is paramount and insufficient pathways effect the timing and operational performance of the final design.

Use of programmable logic in Digital Signal Processing (DSP) applications was also developing and it was apparent that a general purpose logic platform (such as that offered by a CPLD) did not produce an optimum solution. Performance was improved by a employing a hybrid architecture, where conventional general purpose logic sat alongside dedicated blocks; for example Ethernet Media Access Controllers (MACs) and Random Access Memory (RAM). Additionally, the grid-type architecture of CPLDs was supplanted by a more flexible architecture, defined as FPGA. Such an architecture, based on interconnected multi-purpose logic blocks termed Look-Up Tables (LUTs), offered much greater logic density and interconnects [1], [2].

Nowadays, FPGA capability is unparalleled when the performance of specific tasks is compared to those performed using Central Processing Unit (CPU) or Graphics Processing Unit (GPU) [3]. Designs spanning hundreds of millions of transistors are commonplace and complex systems once requiring an entire platform, can be contained within a single device [4]. Typically, FPGA devices include memory, communication blocks and DSP elements, as well as user customisable logic. Such devices have found use in communications systems, control systems, military and automotive applications.

Specifically in communications, the use of FPGA technology is widespread as the operations required for digital-domain modulation and signal processing are well suited for such devices. In particular are methods such as Orthogonal Frequency Division Multiplexing (OFDM) that utilise the Inverse Fast Fourier Transform (IFFT) for the generation of the transmitted signal and the Fast Fourier Transform (FFT) for the detection of the received signal; examples include [5], [6], [7], [8], [9], [10], [11] [12], [13], [14]. Additionally, work on signal diversity methods to increase bandwidth (so called Multiple In Multiple Out (MIMO)), has further driven the use of FPGAs [15], [16], [17], [18]. Predominantly, this is because such applications require parallel processing which is well suited to FPGA implementation.

Recently, effort has been targeted at methods to reduce the bandwidth utilised for spread spectrum signal modulation methods. This is driven by an ongoing need to increase wireless signal availability and capacity, dictated by end-user requirements. It is attractive to maintain compatibility with current methods that include common operations such as the IFFT, so that such advanced methods can easily integrate into current implementation hardware. It is required to offer not just a reduction in bandwidth for a given data-rate, but also a tolerable implementation complexity. Competing methods are detailed and the implementation complexity is discussed in Chapter 5.

1.1 Motivation and Research Aims

FPGA devices and high level hardware development languages represent a new and exciting addition to traditional research tools. Simulation environments, such as Matlab, can be complemented by the direct implementation of complex algorithms and processes. FPGA technology has matured to a point that when diligent coding practices are followed, the resulting logic design has unparalleled performance, in terms of speed and power and can be moved directly to an Application-Specific Integrated Circuit (ASIC) for commercialisation. Designs that are based on well known and long standing standard mathematical operations, such as the FFT, are well suited for

FPGA implementation; such operations have been subject to rigorous optimisation and revision [14], [19], [20].

Implementation of non-standard wireless signal generation methods are generally confined to simulation and modelling, for example Faster-than-Nyquist (FTN) [21], Spectrally Efficient Frequency Division Multiplexing (SEFDM) [22], High-Compaction Multi-Carrier Modulation (HC-MCM) [23] and Overlapping Orthogonal Frequency Division Multiplexing (OvFDM) [24]. Hence, there exists motivation to implement such non-standard techniques into FPGA hardware, so that practical comparisons can be made to existing implementations. Furthermore, simulation environments are not infallible and it is generally preferable to prove the validity of simulations using hardware. At University College London (UCL), research is ongoing in the design, modelling and simulation of the Frequency Division Multiplexing (FDM) techniques (SEFDM) which, for a given data rate, utilise less bandwidth than traditional OFDM methods. Furthermore, SEFDM is mainly based around standard mathematical functions with a smaller subset of non-standard functions and as such is an ideal candidate for FPGA implementation.

The aim of the work is to provide the ability to verify both the theory and the feasibility of SEFDM signal modulation techniques. Such signals are generated by developing FPGA-targeted SEFDM algorithms. The resulting digital SEFDM signal from the FPGA is converted to a real signal using standard Digital to Analogue Converter (DAC) techniques. This signal's time and frequency domain properties can be measured and compared to theoretically and analytically derived values. Furthermore, using standard Analogue To Digital Converter (ADC) techniques, the signal can be used with a decoder capable of accepting the SEFDM encoded signals. Such a methodology can provide verification of SEFDM in an end-to-end configuration. To ensure that the generation of SEFDM is similar to current OFDM standards, such as WiFi [25] and emerging standards such as Long Term Evolution (LTE) [26], the use of FPGA hardware and associated DACs of the highest specification is preferred. In general, platforms containing devices of such high specification are not available and necessitates the design and development of a custom hardware platform. Furthermore, as such an undertaking is complex and time-consuming, consideration is given to additional communication applications, such as MIMO and other broadband methods. By the use of a high specification FPGA based DAC hardware platform and the implementation of SEFDM, the feasibility of SEFDM compared to OFDM in common wireless channels can be performed. Comparisons can be made against published work on SEFDM and competing methods from an implementation perspective. Furthermore, the design can be targeted for ASIC, enabling an additional comparison to currently available Integrated Circuits (ICs) for OFDM generation and confirming the commercial viability of SEFDM.

1.2 Thesis Structure

Chapter 2 details the conceptualisation, design, build and operation for SEFDM signal generation of an FPGA-based platform, with the required circuity to enable digital to analogue signal conversion. The motivation for such an undertaking, in the context of currently available hardware, is discussed and leads to an initial specification for the platform. This specification is extended with reference to currently available ICs which results in a platform block diagram and discussion on the theory of operation. Furthermore, a detailed platform schematic is developed, which in turn is used to produce two Printed Circuit Boards (PCBs). These represent the physical IC interconnects for the required operation. The platform is taken through a staged prototype board bring-up cycle, from power confirmation to bringing up the boards digital and analogue components in a logical manner. Lastly, the FPGA is programmed with test code to confirm the interconnected devices operation; namely Ethernet, RAM and the DACs. Chapter 3 details the platforms validity for use in wireless signal generation by performing analogue characterisation measurement at the output of the DACs, using the FPGA to drive stimulus signals. The types of stimulus and the measurement use is detailed in Table 1.1. Processing of the characterisation is performed by capturing the post DAC signal using a high speed ADC and passing the data to Matlab. Industry standard test equipment is used to confirm the

Table 1.1: Table showing FPGA stimulus to perform characterisation measurements

Stimilus	**Measurement**	**Domain**
Narrow Pulse	Transient	Time
Narrow Pulse	Frequency Response	Frequency
Ramp	Output Linearity	Time
Sine Wave	Noise	Fequency

Matlab processing and results. The characterisation data is used to confirm the correctness of the output signals and where appropriate, to calibrate out any unwanted effects. Chapter 4 details a method of encapsulating the minimum protocol processing elements required for communication over Ethernet. A novel approach, using a pre-defined structure to format the Address Resolution Protocol (ARP) request into a reply, is detailed whereby any other data not designated as ARP is made available for control and reconfigurability operations. Detailed timing and performance results are provided for the FPGA implementation before functional results conclude the design. Chapter 5 gives a detailed account of the design, development and subsequent real signal generation of SEFDM using the hardware platform from Chapter 2, controlled via the Ethernet interface from Chapter 4. Chapter 6 adds a methodology that enables the verification of the generated SEFDM signal by the use of in-the-loop verification. The conclusions and future work are presented in Chapter 7, where the work is summarised and a critical look at it is undertaken. The thesis includes six appendices detailing different practical aspects of the design from Chapter 2 as well as a summary of the Graphical User Interface (GUI), presented as a Datasheet in Appendix D. Furthermore, details on the theory of COordinate Rotation DIgital Computer (CORDIC) and an additional paper (relevant to the work in Chapter 5) written by the author on M-Sequence generation are also included.

1.3 Main Contributions

The work contained in this thesis is on the use of FPGA logic devices and the application thereof in a communications research environment. It details the conceptual through to design, build and test of a custom FPGA based wireless signal generation platform and its verification against vendor specifications. Detailed are novel VHDL sourced logic designs, which enable Ethernet communication and allow for the generation, based on user defined parameters, of a new type of multi-carrier modulation signal; namely SEFDM. Additionally, this work demonstrates a wireless signal verification topology, which directly confirms the literature-based models of SEFDM. The following summarises the author's contributions detailed in the thesis chapters:

- Proposed, designed and implemented a complete FPGA based hardware system for the purpose of broadband wireless signal generation and transmission. As of the date of implementation (2007), this system represented the state of the art in terms of FPGA connectivity to concatenated DAC devices with total analogue signal bandwidth of up to 1.2GHz. The system design requires highly specialised knowledge of high speed digital PCB design, memory and Ethernet interfaces and mixed signal power and layout combined with FPGA design methods.

- Designed and implemented a self testing capability integrated within the FPGA architecture, for verification and testing to ensure that IC vendor specification and performance criteria are achieved.

- Developed a new efficient method to enable Ethernet connectivity from a host computer to the FPGA, removing the need for the traditional software stack to manage standard Ethernet communication protocols. This solution is shown to have a smaller resource footprint relative to standard solutions and has applications beyond the communication system for which it was designed.

- Derived a modified mathematical description of SEFDM signals suitable for practical hardware implementation. This was verified using VHDL code targeted for FPGAs and then was used for the generation of various SEFDM signals and extended the system capabilities by allowing parametrisation using data supplied from a host computer over Ethernet. This led to a versatile design with reconfigurability of the FPGA logic to produce signals with different spectral properties.

- Designed and implemented a user friendly GUI for the control of the designed FPGA system.

- Developed and implemented a new verification technique based on the use of a newly custom developed *in the loop* methodology. This enables data integrity testing and assessment of the generated SEFDM signals. Furthermore, with real signal capture and the use of a model detector, the methodology enables system performance assessment in realistic channel environments.

- Critically evaluated the different SEFDM implementation methods, from a viewpoint of complexity growth. Conclusions are drawn regarding the appropriate method which provides the most resource efficient FPGA design based on system engineering parameters, such as number of carriers, the bandwidth saving required and the granularity in frequency of each sub-carrier when a reconfigurable system is considered.

1.4 List of Publications

This work has resulted in seven conference papers and one invited IEEE journal paper listed below in chronological order:

1. Perrett, M. and Darwazeh, I. *FPGA Implementation of Quad Output Generator for Spectrally Efficient Wireless FDM Evaluation*, International Symposium of Broadband Communications (ISBC'2010), Melaka, Malaysia.

2. Whatmough, P., Perrett, M., Isam, S. and Darwazeh, I. *VLSI Architecture for a Reconfigurable Spectrally Efficient FDM Baseband Transmitter*, IEEE International Symposium on Circuits and Systems (ISCAS'2010), Rio de Janeiro, Brazil.

3. Perrett, M. and Darwazeh, I. *Flexible Hardware Architecture of SEFDM Transmitters with Real-Time Non-Orthogonal Adjustment*, IEEE International Conference on Telecommunications (ICT'2011), Ayia Napa, Cyprus.

4. Perrett, M. and Darwazeh, I. *Verification of FPGA Generated SEFDM Signals*, London Communication Symposium (LCS'2011), London, UK.

5. Perrett, M. and Darwazeh, I. *A Simple Ethernet Stack Implementation in VHDL to Enable FPGA logic reconfigurability*, IEEE International Conference on Reconfigurable computing (ReConfig'2011), Cancun, Mexico.

6. Perrett, M., Grammenos, R. and Darwazeh, I. *A Verification Methodology for the Detection of Spectrally Efficient FDM Signals Generated using Reconfigurable Hardware*, IEEE International Conference on Communications (ICC'2012), Montreal, Canada.

7. Whatmough, P., Perrett, M. Isam, S. and Darwazeh, I. *VLSI Architecture for a Reconfigurable Spectrally Efficient FDM Baseband Transmitter*, IEEE Transactions on Circuits and Systems (TCAS-2), (invited paper) May 2012.

8. Perrett, M. and Darwazeh, I. *Characterisation and verification of an FPGA signal generator for spectrally efficient wireless FDM*, IEEE International Conference on Telecommunications (ICT'2012), Beirut, Lebanon.

Chapter 2

The Design and Build of an FPGA-Based Signal Generator

Publications relevant to this chapter are detailed as follows:

Perrett, M. and Darwazeh, I. *FPGA Implementation of Quad Output Generator for Spectrally Efficient Wireless FDM Evaluation*, International Symposium of Broadband Communications (ISBC'2010), Melaka, Malaysia.

Modern wireless signal generation is predominately based on digital preprocessing and the subsequent conversion into analogue signals, using Digital to Analogue Converters (DACs) [10], [27]. It is commonplace to encapsulate symbol mapping, modulation, equalisation, coding, scrambling, phase alignment and pulse shaping techniques digitally. These are often implemented in devices such as Application-Specific Integrated Circuits (ASICs) and Field Programmable Gate Arrays (FPGAs). There is significant interest in targeting analytically proven algorithms to FPGAs [28], [29], [30]; realising performance improvements offered by such technologies [3], [31]. For research purposes, FPGA vendors offer evaluation boards for the latest generation of devices. These boards are rarely available in conjunction with the latest generation of DACs or Analogue To Digital Converters (ADCs). Therefore, investigation into algorithms which require both fast algorithmic computation, wide bandwidth and high precision analogue conversion are confined to analytical evaluation, using environments such as Matlab.

Matlab evaluation is dependent on simulation parameters provided by the programmer; so called *modelling*. Nowadays, such models describe highly complex systems and are often accepted as proof of correct operation. However, simulation cannot replace practical experimentation as unknown effects still exist. Hence, there exists a requirement for researchers to have access to current generation programmable hardware and signal converter methods, to prove the viability of proposed techniques practically. Unfortunately, such platforms are not readily available and require development, targeted specifically at the evaluation of the required algorithm. This chapter describes such a system for the purpose of evaluating Frequency Division Multiplexing (FDM)-type signals, where the sub-carrier spacing is compressed to reduce the bandwidth required.

Typically, IC vendors will release an application development platform alongside a new design release. This is to enable potential customers to evaluate the design without the need to embark on expensive and time consuming board customisation and development. Vendor platforms are limited to the evaluation of the device itself and generally will not include a capability to interface to other devices from other vendors. If this capability is included, it is generally via sub-optimal interfaces that do not allow full specification evaluation. Hence, if the full potential of devices from different vendors were to be evaluated together, a customised solution is required, connected in an optimum manner.

Hitherto, development is undertaken to provide a bespoke platform for wireless signal generation. The aim is to combine a high performance programmable logic device (an FPGA), together with a high performance DAC to create a wireless signal generator. The purpose is twofold - to provide high performance logic capability and to provide analogue conversion capability. This enables the real time implementation of signal generation methods, which are currently considered too complex for main-stream use [32], [33]. Additionally, due to current interest in Multiple In Multiple Out (MIMO) wireless architectures [34], [18], [15], it is proposed that four DACs are utilised. The FPGA will control data throughput and perform data processing, before outputting data to each DAC. It is also required to have user interaction for operational control and to provide source data for processing. Therefore, a standard Personal Computer (PC) interface is also implemented. As common PC interfaces, such as Universal Serial Bus (USB) 2.0 or Gigabit Ethernet (GbE), have insufficient data bandwidth to provide data in real time to the DAC, some memory is required for intermediary storage of data before it is processed and output to the DACs at full speed.

Section 2.1 summarises the current (as of 2007) state of the art. This review is used to generate a specification, which is detailed in Section 2.2. This specification is based on the highest performing parts available. The operation of the proposed system is provided in Section 2.3, where the functionality and interoperability of the components selected is detailed. Section 2.4 further elucidates individual areas of the design; namely Ethernet, the FPGA, the DACs and memory. Furthermore, the subsequent power requirements are analysed and a power strategy is devised. A schematic of component interconnects is developed in Section 2.5, before a study into the requirements and design detail of a Printed Circuit Board (PCB) is provided in Section 2.6. Section 2.7 details the method of placing the required components onto the PCB, before section 2.8 provides a detailed description of the procedure used to test preliminary functions of the system in isolation. It also details issues that are found and remedial action taken. The system and its resultant performance is concluded in Section 2.9.

2.1 The State of the Art

Common DAC vendors include Maxim [35], Analog Devices [36], Texas Instruments [37], NXP [38] and Linear Technology [39]. A comparison of all the products offered by each of these vendors is not feasible within the scope of this thesis. However, it is noteworthy that Analog devices offer the fastest, high bit-resolution parts (16-bit at 300MHz) which best suites the requirements for the highest performing system components. The two major vendors of FPGA devices are Xilinx [40] and Altera [41]. Other manufacturers exist that specialise in specific areas, such a Lattice [42] and Actel [43]. Table 2.1 summarises the various FPGA vendors. Xilinx and Altera

Table 2.1: Table comparing programmable logic device manufacturers available in 2007

Metric	**Xilinx**	**Altera**	**Lattice**	**Actel**
Performance	High	Mid-high	Low	Very Low
Power	Med	Low	Med	Very Low
Cost	High	high	Med	Low
Built-in Flash	N/A	N/A	Yes	Yes
CPLD	Med	Very Low	High	Med
Mixed Signal	N/A	N/A	N/A	High

represent the cutting edge for performance and power utilisation. Both are due to advanced internal architectures and use of leading edge lithography for production. Lattice has the highest performing Complex Programmable Logic Device (CPLD) device, but CPLDs is a mature technology which is difficult to scale and cannot compete with FPGA technology. Actel focus on specialist low power or mixed signal FPGA and CPLD devices. Nonetheless, Xilinx and Altera are the most widely used and best supported, as well as the largest (in terms of resource) and highest performing (in terms of clock frequency and internal routing). Additionally, Xilinx are the only vendor to offer free-of-charge Ethernet interfaces embedded into the FPGA.

Memory parts using Double Data Rate (DDR) technology are available with up to 1Gb per device, such as Micron MT46H32M32 [44], IDT DT71P71804 [45], GSI GS8662T36 [46] and Cypress CY7C1510V18 [47]. Importantly, part availability is challenging if small quantities are sought. Therefore, a trade-off must be considered between specification and part availability. Currently (as of 2007), the Micron parts have the largest capacity which is available in small quantities from distributors.

2.2 Specification

Based on the highest specification devices currently available, a broad specification is developed as follows:

- Analogue Bandwidth of $150MHz$
- Signal to Noise Ratio (SNR) $\geq 90dB$
- Up-conversion = 250MHz to 3GHz
- 1Gb memory storage per DAC
- Gigabit Ethernet user interface

The capacity of the FPGA is subjectively based on the optimisation effort during design. Therefore, different parameters are considered in Section 2.4.2 to determine the FPGA device required.

2.3 Operation

This section details the general operation of the system. Conceptually, the principle aim of the system is to process data, supplied from an Ethernet interface, into a format suitable for wireless transmission. Subsequently, such data is converted into an analogue signal using standard techniques and made available for transmission over a channel. This procedure is detailed diagrammatically in 2.1. In detail, data supplied from a user using a custom developed Graphical User Interface (GUI) is transmitted over Ethernet to the FPGA. Before this data can be accessed, the FPGA must perform the minimum tasks required for data transmission over Ethernet (as detailed in Chapter 4). Once the data is available to the FPGA, it is routed to an attached Random Access Memory (RAM). Control signals, also over Ethernet, instruct the FPGA which RAM the data is destined for; there is a RAM per DAC. Once the data download from the user is complete, a control signal determines what action shall be performed by the FPGA as follows:

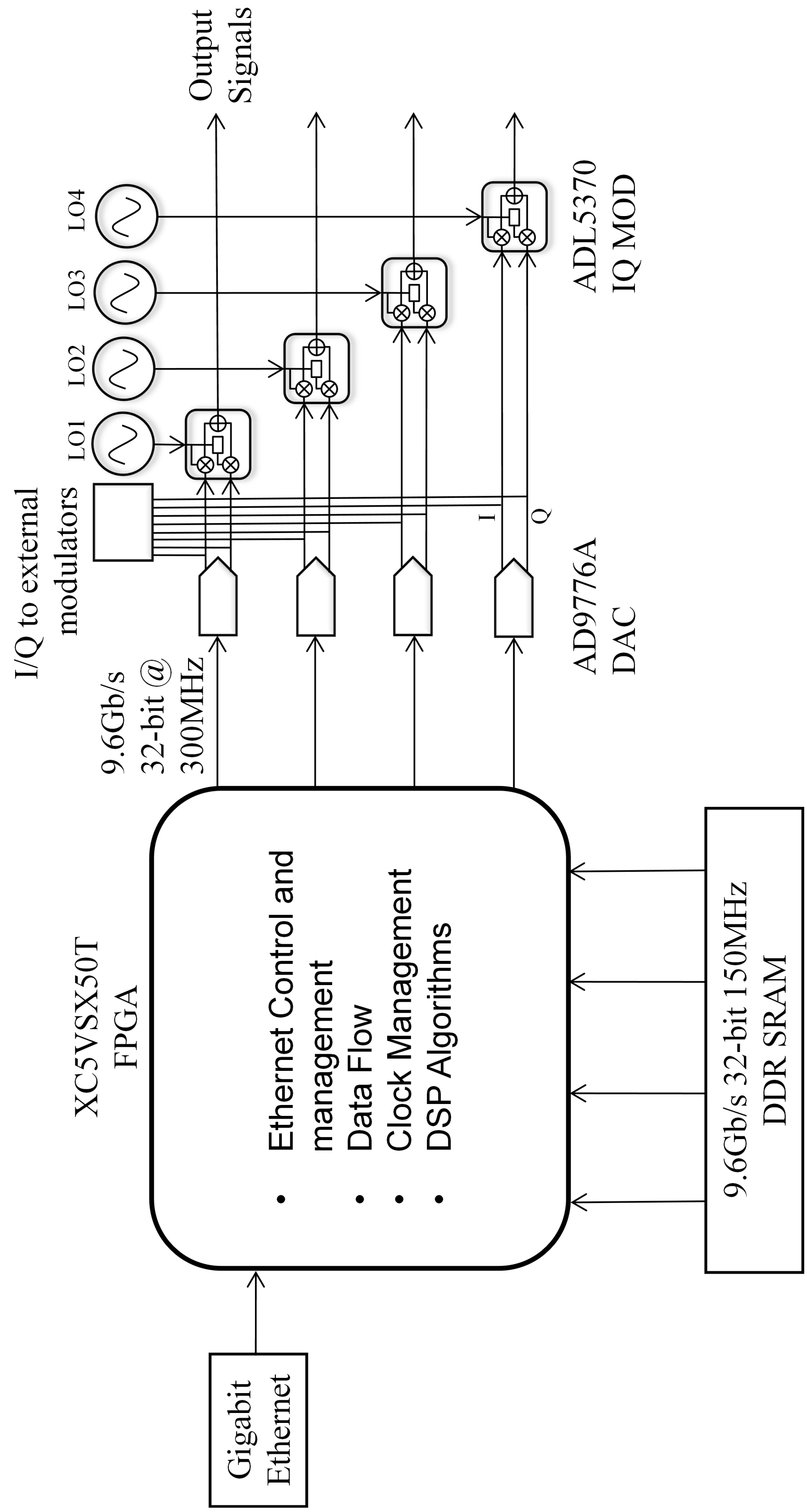

Figure 2.1: Conceptual system block diagram

- Load the data from the RAM and output it to the corresponding DAC without any processing applied.

- Load the data from the RAM and apply a pre-defined operation. For example, modulate the data using Orthogonal Frequency Division Multiplexing (OFDM) techniques.

- Use the data supplied to perform debug or characterisation operations, internal or external to the FPGA.

The FPGA performs the task required via pre-configured logic, which is detailed in Chapter 5. Regardless of the action taken by the FPGA, the method of data output to the DAC is common. This involves transforming the data to a suitable format for the DAC, which is transmitted at a rate determined by the DAC clock. Each DAC has an independent I and Q channel, so data must be formatted appropriately by the FPGA. At the DAC output, the resulting I and Q signals can be measured directly by passing through a buffer amplifier. Alternatively, the I and Q can be combined together using an on-board IQ modulator. The resulting frequency is determined by an externally sourced local oscillator, which the IQ modulator uses to produce a signal suitable for wireless transmission.

2.4 Component Selection

This section provides greater detail of specific areas of the system. The sections elucidated are Ethernet (Section 2.4.1), the FPGA (Section 2.4.2), DACs (Section 2.4.3) and on-board memory (Section 2.4.4). The result leads to a summary of the major components to be used in Section 2.4.5. Additionally, the Power requirements are analysed and a power strategy devised in Section 2.4.6 followed by general design highlights in Section 2.4.7. Such highlights include reset strategy, decoupling and off-board connectors.

2.4.1 Ethernet Hardware Interface

A typical Ethernet interface consists of three blocks; namely the Physical interface, the PHYCeiver and the Media Access Controller (MAC). Historically, a MAC was necessary to manage two-way packet transaction over a single cable interface (defined as half-duplex). Due to the high possibility of simultaneous node packet transmission, the MAC implemented a method of two-way packet management, known as Carrier Sense Multiple Access / Collision Detection (CSMA/CD). In detail, Carrier Sense (CS) allowed the connected MAC nodes to *listen* for quiet periods (where no communication was occurring), before Multiple Access (MA) subsequently allowed each node to initiate a data transaction. Each node has equal opportunity to initiate a transmission and Collision Detect (CD) managed the detection of two nodes communicating simultaneously. Any subsequent packet re-transmission required is based on random time periods, so called *back-off* time. In modern full-duplex connections, the MAC is responsible for issuing pause commands to a connected host if the receive buffers are saturated and packet loss will otherwise commence. Furthermore, the MAC is responsible for performing data integrity checking per Ethernet frame. The PHYCeiver operation is broken down into multiple stages as illustrated in Figure 2.2. The physical interface, commonly implemented using an RJ45 connector, provides

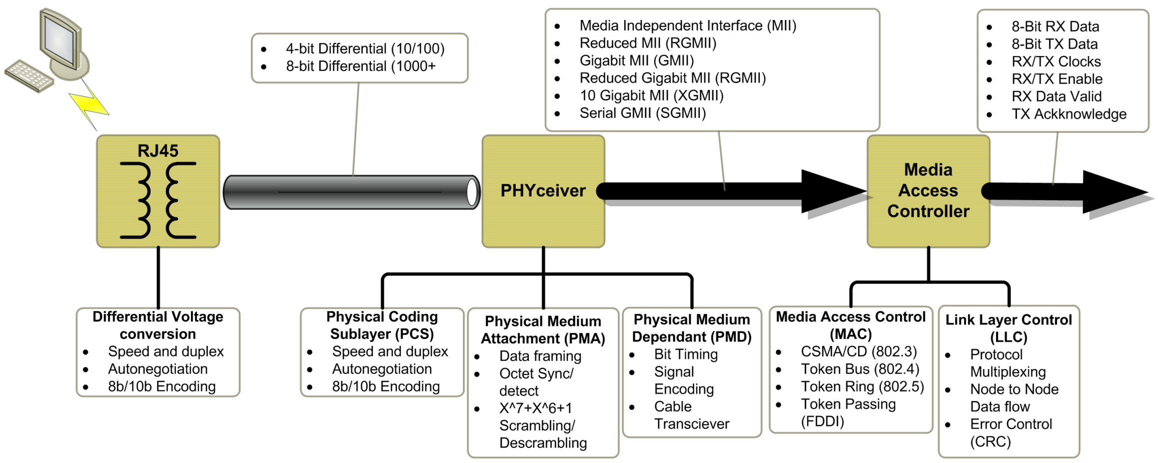

Figure 2.2: Block Diagram of Ethernet Physical and MAC

line-isolation as defined in IEEE 802.3ab Clause 40.8. The maximum target interface (100 Base-TX for example) determines the number of differential pair data pins required. A circuit diagram of a typical physical interface is shown in 2.3. In this

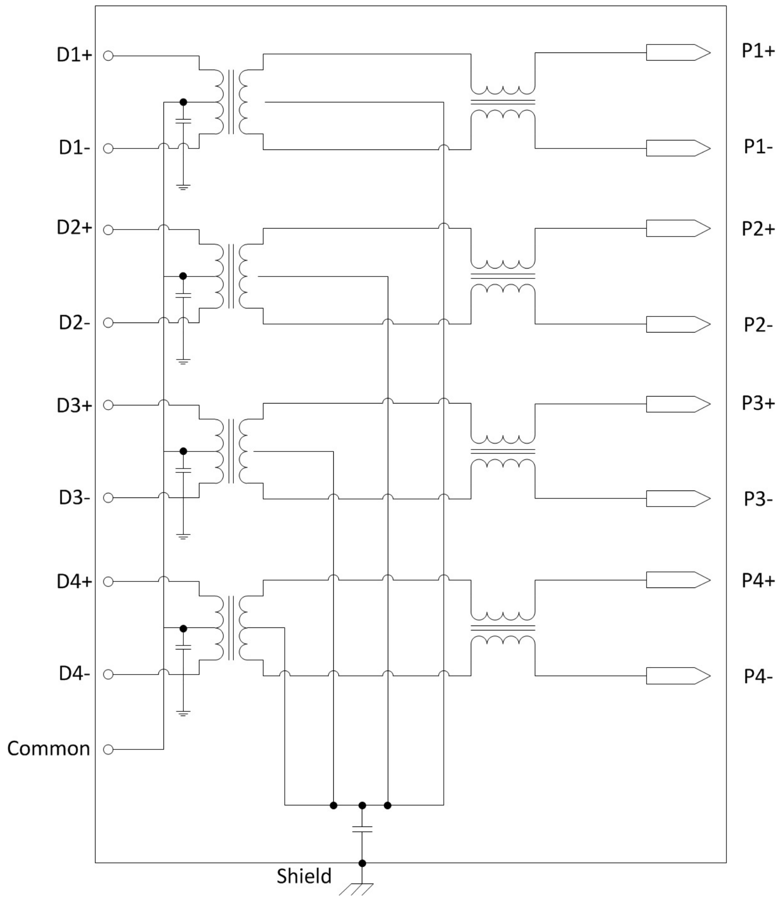

Figure 2.3: Circuit Diagram of a Gigabit Ethernet RJ45 connector showing the voltage transformer and differential data carrying pairs

example, there are four sets of differential I/Os, each with a separate transformer. Additionally, there is a common tap that is connected to an Electro-Magnetic Interference (EMI) shield that surrounds the connector. This configuration is commonly utilised in Gigabit Ethernet applications. A PHYCeiver comprises of 3 separate sub-blocks, each with an individual subset of services or features. The first block (Physical Coding Sub-Layer (PCS)), is responsible for determining speed, duplex mode (half or full) and line-coding. Speed concerns the number of bits per second which are transferred over the network, while duplex determines if read and write operations can occur simultaneously. The second block (Physical Medium Attachment (PMA)) manages data framing (the detection of the start preamble which signifies the beginning of a new Ethernet frame), octet synchronisation (the realignment of 4-bit nibbles to an 8-bit bytes) and scrambling operations. Scrambling is used to decorrelate the transmitted signal to reduce interference. The third block (Physical Medium Dependant (PMD)) is where bit-level timing and the MAC interface are situated. Such interfaces can vary depending on data-rate, power budget, available number of pins and available interfaces, which are summarised in Table 2.2: It is obvious that

Table 2.2: Interconnect standards available for data transfer between a PHYCeiver and MAC

Interface	**Abbreviation**	**Pins**
Medium Independent Interface	MII	16
Gigabit Medium Independent Interface	GMII	26
Reduced Medium Independent Interface	RMII	10
Reduced Gigabit Medium Independent Interface	RGMII	12
Serial Gigabit Medium Independent Interface	SGMII	8

latter standards (reduced and serial) require less physical pins, but at the expense of either a greater complexity (for reduced) or specific serial hardware requirements (for serial), such as serial transceivers. For this design, the Reduced Gigabit Media Independent Interface (RGMII) standard is chosen as it provides a significant reduction in pins compared the Gigabit Media Independent Interface (GMII), but without requiring resource costly serial transceivers for using Serial Gigabit Media Independent Interface (SGMII) [48].

2.4.2 Programmable Logic Device

The platform will serve as a research test-bed for algorithmic evaluation. Therefore, it is preferable that the highest performing programmable logic device possible is implemented. FPGAs can be measured in terms of speed (the clock frequency can be tolerated without error), peripherals (built in hard logic such as Ethernet and Digital Signal Processing (DSP)) and resource (the size of the logic that be programmed). The author has had previous experience with Xilinx FPGAs and design tools. Additionally, Xilinx offer built in Ethernet blocks [49] that reduces the resource requirement if used. As the design will utilise Ethernet, Xilinx is selected. Furthermore, as the FPGA will be used for algorithms, a variant with larger DSP blocks is preferred. A breakdown of the available Virtex 5 parts from the FPGA data-sheet in displayed in Figure 2.4 It can be seen that the LX series does not have any embedded Ether-

Device	Configurable Logic Blocks (CLBs)			DSP48E Slices(2)	Block RAM Blocks			CMTs(4)	PCI Express Endpoint Blocks	Ethernet MAC Blocks	Maximum RocketIO GTP Transceivers(5)	Total I/O Banks(7)	Max User I/O(6)
	Array (Row x Col)	Virtex-5 Slices(1)	Max Distributed RAM (Kb)		18 Kb(3)	36 Kb	Max (Kb)						
XC5VLX30	80 x 30	4,800	320	32	64	32	1,152	2	N/A	N/A	N/A	13	400
XC5VLX50	120 x 30	7,200	480	48	96	48	1,728	6	N/A	N/A	N/A	17	560
XC5VLX85	120 x 54	12,960	840	48	192	96	3,456	6	N/A	N/A	N/A	17	560
XC5VLX110	160 x 54	17,280	1,120	64	256	128	4,608	6	N/A	N/A	N/A	23	800
XC5VLX220	160 x 108	34,560	2,280	128	384	192	6,912	6	N/A	N/A	N/A	23	800
XC5VLX330	240 x 108	51,840	3,420	192	576	288	10,368	6	N/A	N/A	N/A	33	1,200
XC5VLX30T	80 x 30	4,800	320	32	72	36	1,296	2	1	4	8	12	360
XC5VLX50T	120 x 30	7,200	480	48	120	60	2,160	6	1	4	12	15	480
XC5VLX85T	120 x 54	12,960	840	48	216	108	3,888	6	1	4	12	15	480
XC5VLX110T	160 x 54	17,280	1,120	64	296	148	5,328	6	1	4	16	20	680
XC5VLX220T	160 x 108	34,560	2,280	128	424	212	7,632	6	1	4	16	20	680
XC5VLX330T	240 x 108	51,840	3,420	192	648	324	11,664	6	1	4	24	27	960
XC5VSX35T	80 x 34	5,440	520	192	168	84	3,024	2	1	4	8	12	360
XC5VSX50T	120 x 34	8,160	780	288	264	132	4,752	6	1	4	12	15	480
XC5VSX95T	160 x 46	14,720	1,520	640	488	244	8,784	6	1	4	16	19	640

Figure 2.4: Figure showing the Xilinx Virtex 5 family breakdown (from vendor datasheet)

net blocks and is not suitable for the application. The LXT parts have embedded Ethernet blocks, but a comparison of the LX50TG and SX50T reveals the SX50T to be superior in other important areas; namely block RAM and DSP slices. The cost of larger parts, (for example to a LX85T or SX95T) is prohibitively expensive *. Therefore, the device selected, based on highest performing within cost constraints, is the Virtex SX50T. This device has a balance of cost, performance, embedded logic and DSP blocks.

All FPGAs have flexible pin allocations, with each pin capable of performing multiple functions [50]. However, there are special pins dedicated to clocks that have tight design guidelines attached. Importantly, the use of an input buffer and subsequent clock tree buffer are required to enable the best timing results. Such a configuration is illustrated in Figure 2.5. Most high speed oscillators are transmitted using differential pairs. Generally the pair of signals must be terminated at the destination by a 100Ohm resister, as seen on the left part of Figure 2.5. For Xilinx FPGAs, such

*The largest devices, in small quantities (1-2 devices), can cost in excess of £5000 each

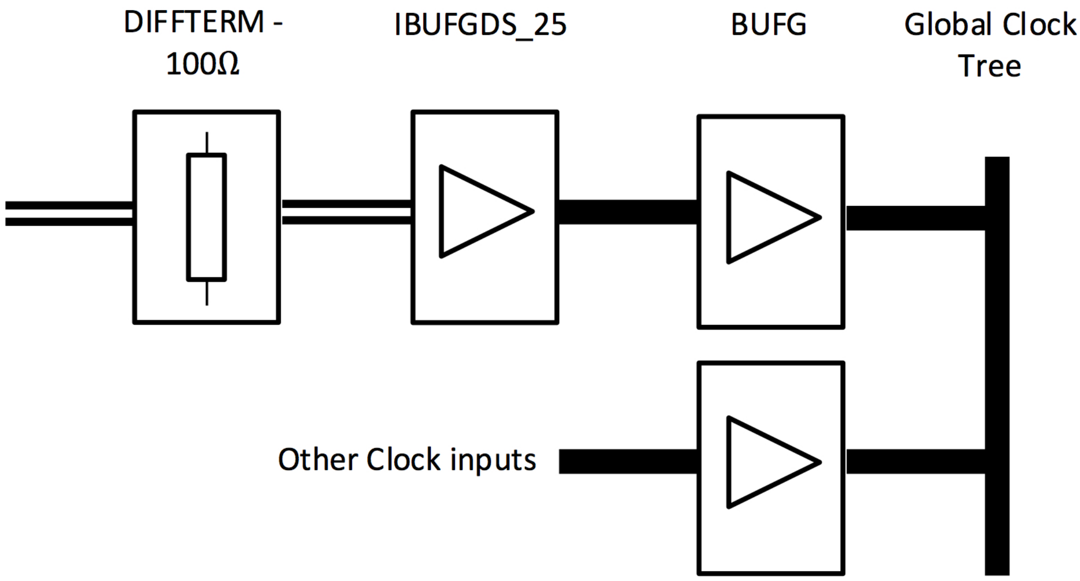

Figure 2.5: Block diagram of standard clock interface requirements from pin to internal clock tree (parallel lines are differential signals)

resistors are integral to the FPGA fabric and are enabled when a differential pair is used at design time. Internal to the FPGA, the differential pair is input to an IBUFGDS [51]. This is both a buffer and dual to single ended converter. To ensure this clock has optimum routing and performance inside the FPGA, a BUFG is used to gain access to the global clock tree. The BUFG is a hard-coded logic block with a defined timing profile. Following this guideline is paramount if a high performing and resource efficient design is sought.

Furthermore, the program within an FPGA does not persist once source power is turned off. Instead, a separate flash storage device is required to hold the program. The FPGA has a hard-coded interface specifically for retrieving a program from such external flash devices. The process of flash program retrieval is instigated when power is first applied. The interface between the FPGA and the flash device is specified in the Xilinx configuration guide [51] and is exemplified in Figure 2.6. The flash is programmed using a Joint Test Action Group (JTAG) interface, the details of which are provided in later sections during its use to test the preliminary functions of the system.

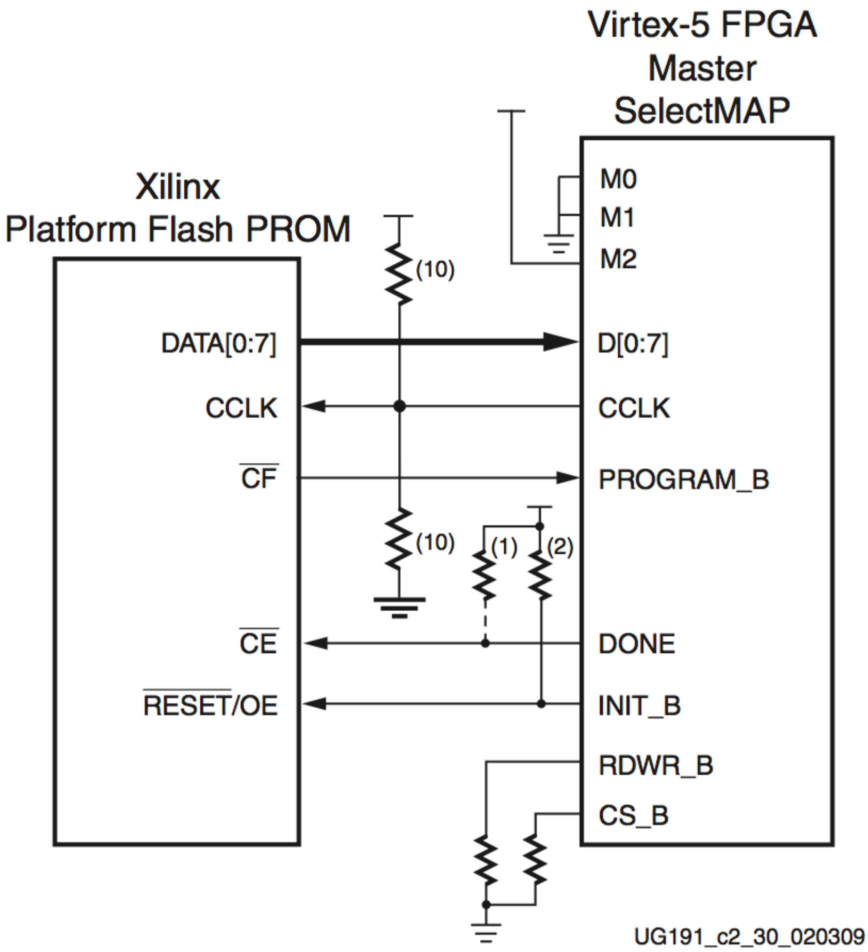

Figure 2.6: Figure showing a typical connection between an FPGA and an external flash memory (from vendor datasheet)

2.4.3 Digital to Analogue Conversion

Integrated Circuit (IC) devices are readily available which provide a complete digital to analogue solution. Such devices have digital inputs, clocking, phase correction, analogue line balancing and DSP features included. Performance of these devices is well characterised based on two metrics, the sampling rate and the digital input bit-width. Sampling rate corresponds to the available analogue bandwidth at the output, while the bit-width of the input data sets an upper bound on the maximum achievable SNR [52]. These parameters are summarised in Figure 2.7.

The appropriate type for the chosen application is a DAC with the highest possible

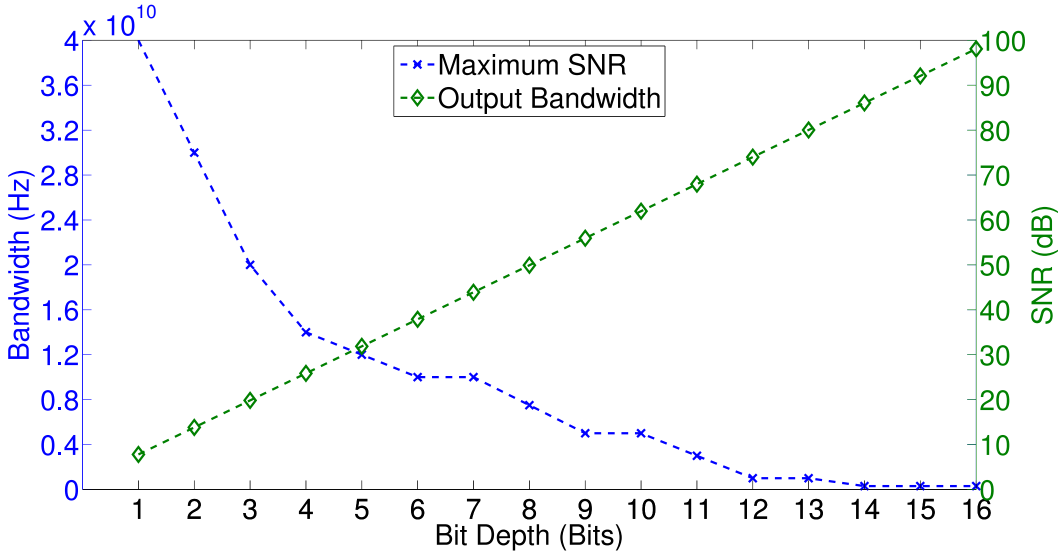

Figure 2.7: Figure showing SNR for increasing bit-resolution (green) with a corresponding reduction in available bandwidth (blue)

SNR performance. As of 2007, the top performing device with the highest SNR performance is the Analogue Devices AD977x series. These devices offer up to $300MHz$ sampling rates with bit-widths up to 16-bit. The AD977x series comprises of two independent DACs per device; one for the I channel and one for the Q channel. The basic operation is illustrated in Figure 2.8, taken from the vendor data-sheet [53]. Principally, these devices are designed to connect directly to two I/Q modulators

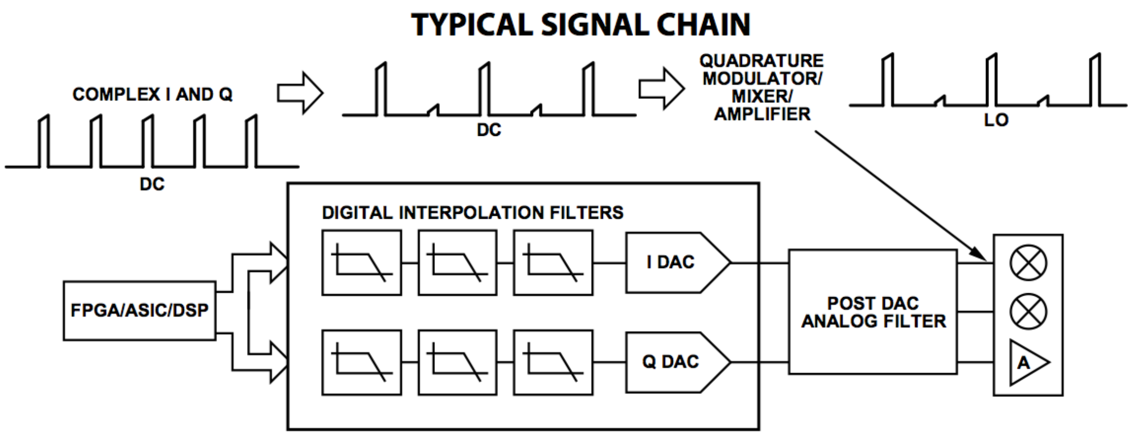

Figure 2.8: Figure showing the basic operation of the AD9779 (from vendor datasheet)

from the same vendor [54]. Such modulators perform up-conversion and transmission, using an external local oscillator, in a configuration as in Figure 2.9. As both

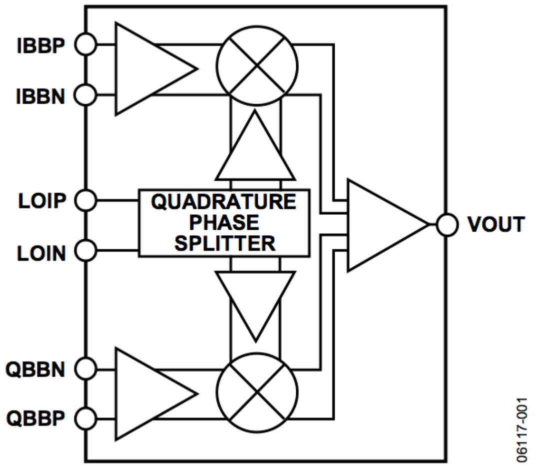

Figure 2.9: Figure showing the IQ modulator external connections and internal function (from vendor datasheet)

parts are from the same vendor, connecting the two devices together is straightforward. The process requires some external components between the DAC and the IQ modulator. These components provide phase feedback (not implemented in the system), line balancing and signal coupling. A typical configuration from the vendor data-sheet is illustrated in Figure 2.10. In the system proposed, it is required that

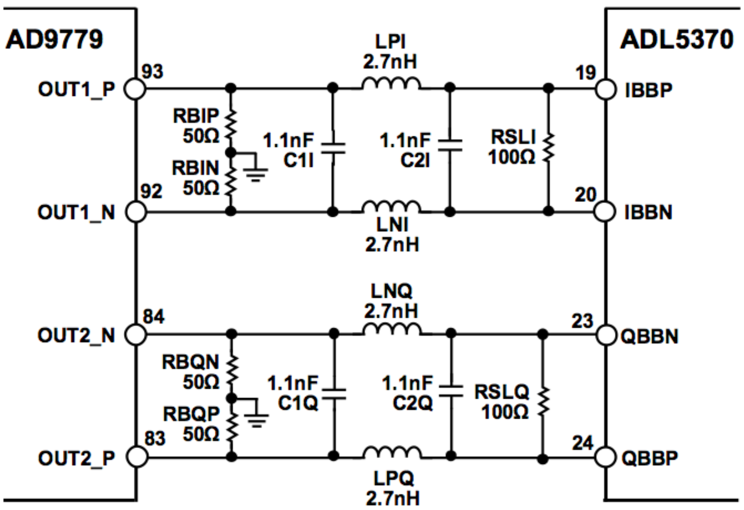

Figure 2.10: Figure showing a typical connection between the DAC and the IQ modulators (from vendor datasheet)

the independent I and Q are also available for measurement without up-conversion. To satisfy this requirement, a buffer amplifier is connected to each I and Q channel in parallel to the up-converter. This buffer has as high an input impedance as possible to ensure signal integrity between the DAC and the up-converter is not adversely impacted. This configuration is illustrated diagrammatically in Figure 2.11. The buffer

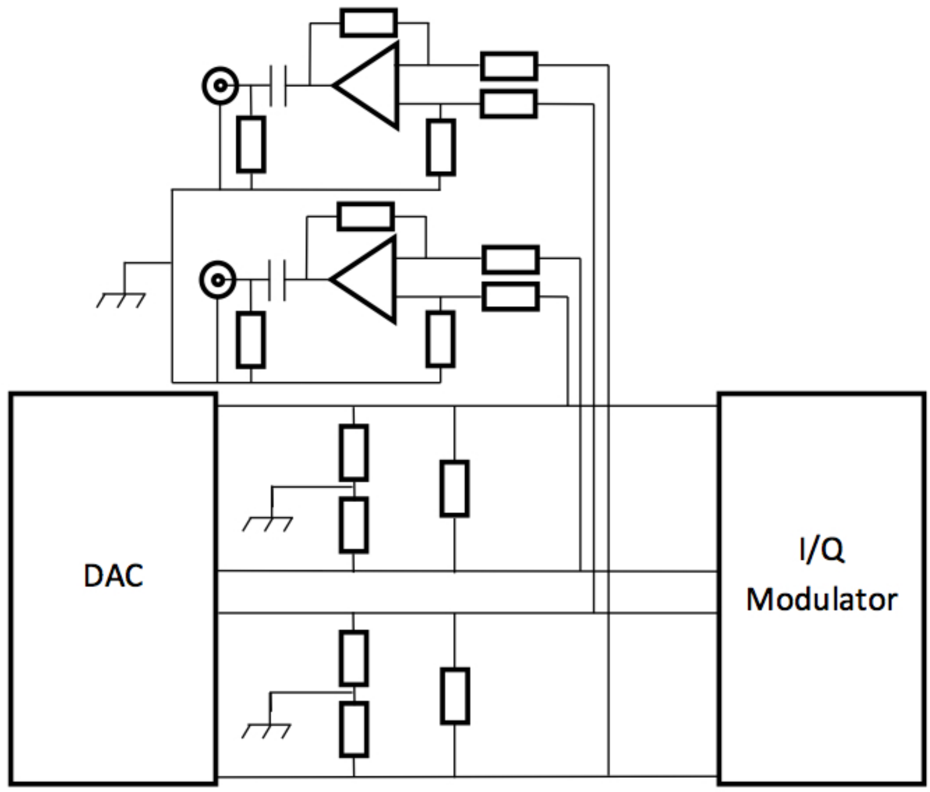

Figure 2.11: Block diagram showing the DAC output connection to the IQ modulator and buffer amplifiers

amplifier requires a suitable output bandwidth to match that of the DAC. Amplifiers available at the time are capable of up to 1GHz operation and the device selected is an Analogue Devices AD8003 [55]. The required operation of the op-amp is in a buffer configuration. When such a configuration is analysed, it is found that the positive and negative inputs are unbalanced. This can be mitigated by the use of suitably high values of resistors. However, the AD8003 can only meet the required bandwidth when low (for example 100Ohm) resistors are used. Hence, there is a trade-off between input balance and output bandwidth. A compromise is to select the minimum allowable bandwidth (i.e the maximum signal bandwidth required and not the potential DAC bandwidth) and utilise the highest values of resisters to minimise input unbalancing. It is expected that this decision will have an adverse effect of the performance, which will be quantified in later Chapters.

The DAC uses 16-bits to represent an analogue voltage, and requires data in 2's complement at its input. The range 8000_H to $FFFF_H$ is the negative output and 0000_H to $7FFF_H$ is the positive output. The output voltage to input value translation is illustrated in Figure 2.12.

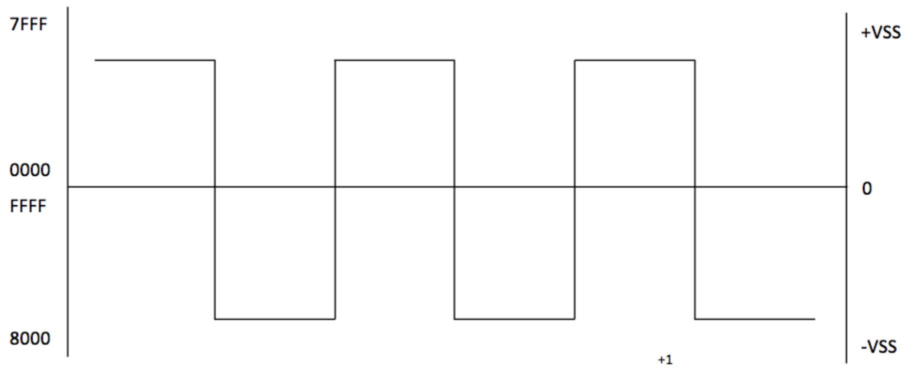

Figure 2.12: Figure showing the 2's complement input of the DAC and the corresponding output voltage

The maximum output signals' effective bandwidth is half the maximum DAC frequency. A translation from digital to analogue happens every rising clock edge. Therefore, if the polarity of data applied to the device is swapped at $300MHz$ rate, a signal with an $150MHz$ bandwidth is output. The data rate required to the DAC at full rate is broken down as follows: Each DAC has dual 16-bit data interfaces (one for I and one for Q). If this data is clocked at a rate of $300MHz$, each DAC requires $16 * 300 * 10^6 * 2$ bits per second, which is $9.6Gb/s$. There are four such DAC devices and the total data throughput required to enable the maximum performance of all devices is $4 * 9.6 * 10^9$. This results in a total data throughput requirement of $38.4Gb/s$. This value is higher than than the data-rate offered by GbE (1Gigabit (Gb)/s). Therefore, such data will be downloaded at a slower rate over Ethernet and buffered locally in memory. The data can then be read by the FPGA and be processed and output at the required DAC rate.

2.4.4 Memory

As previously outlined, some on-board memory attached to the FPGA is required to provide intermediary buffering between the slow user interface and the high speed DAC interface. The size of the memory determines the length in time of each sequence. If Bit Error Rate (BER) measurements are required, these usually require large data-sets to produce valid results. For small (10MB) RAM requirements, external Static Random Access Memory (SRAM) can be utilised. This type of memory provides independent address, clock and data buses and the interface specification is trivial. However, this type of memory does not offer large storage such as that required for the aforementioned BER measurements.

Dynamic Random Access Memory (DRAM) memory in DDR form offers a much larger capacity at the expense of interface complexity. The largest RAM of this type available (as of 2007) that can operate at the required data-rate is 1Gb; organised as 32-bits wide by $32 * 10^6$ deep. DDR memory interfaces are generated using FPGA vendor tools as the requirements are complex and timing sensitive. For this system, it is required that each DAC has dedicated DDR memory, which necessitates four such interface blocks. The Xilinx Memory Interface Generator (MIG) tool [56] can automatically generate interface logic for a chosen DDR memory. However, a limited number of interface and device types can support multiple memory controllers. The logic device selected (Xilinx Virtex XCV5SX50T) cannot support multiple memory interfaces. To circumvent this limitation the following method is utilised, which results in 4 separate instances of the same MIG generated controller. These steps are required to get the MIG to place the physical pins automatically for all the controllers, but further modifications at the implementation stage are also required.

1. Generate a single DDR memory controller to the required specification and target device.

2. Allow the tools to set the pins and generate the interface.

3. Implement the MIG generated footprint allocation in the main project.

4. Generate another interface using the same specifications.

5. Force the tools to use the previously generated footprint file at the pin allocation stage.

6. Repeat steps 3 to 5 until all 4 interfaces have been generated and pins allocated.

Additionally, each MIG controller generates its own IDELAYCTRL [51] and differential clock to single ended clock buffer. In Xilinx devices, IDELAYCTRL is handled in two different ways. If a single IDELAYCTRL is instantiated, the vendor tools automatically spawn this instance over the entire device as a single delay. If multiple IDELAYCTRLs are instantiated (such as when the process described above is utilised), the tools require these to be placed in specific regions of the device. Usually, the MIG would place these when the controller is generated. However, here this is not possible and these must be placed by the user. Without knowledge of which FPGA slice relates to which bank of IO and internal logic, this is very demanding task. To solve this problem, multiple IDELAYCTRL instances are removed from the MIG generated files. Instead, these instances are brought into higher level design files. A single IDELAYCTRL is generated and the output passed to the multiple MIG instances. Since there is now only a single IDELAYCTRL, the tools will automatically spawn the delay throughout the device. Some limitations are present due to the non-standard use of the MIG to produce four independent interfaces. Due to the single IDELAYCTRL, the same clock must be used for each RAM. Therefore, each RAM will operate at the same speed. Furthermore, any other instance that requires an IDELAYCTRL, will need modification to suit the single instance.

2.4.5 Component Summary

Hitherto, the components selected for the system are summarised in Table 2.3. Included is the part number and the footprint which will be used during the PCB design phase. Each device has specific power requirements, but some commonality exists. Therefore, as part of the power strategy such commonalities are taken into account

Table 2.3: Table detailing the component selection for the major operations of the system

Component	Vendor	Part	Pinout	Ref
FPGA	Xilinx	XCV5SX50T	FF1136	[57]
DAC	Analog Devices	AD9779	TQFP-100	[53]
IQ Mod	Analog Devices	AD5370	LFCSP-24	[54]
Buffer Amp	Analog Devices	AD8003	LFCSP-24	[55]
Ethernet PHY	National Semiconductor	DP83865	PQFP-128	[58]
RAM	Micron	MT46H32M32	168-ball BGA	[59]

and combined together. However, mixed signal or domain interdependent devices will need dedicated power sources even for the same voltage. Section 2.4.6 provides a study into the power requirements of the devices from Table 2.3.

2.4.6 Power

The design demands multiple voltages over digital and analogue domains. The power requirements of most devices can be found from the vendor specification. However, the power requirement of the FPGA can change with the logic that is downloaded. Vendor tools provide an estimation of the power requirement based on user defined code [60], the result for this design can be found in Figure 2.13. The code used as the source for this estimate is detailed in later sections. The requirements specified

Name	Power (W)	Voltage	Range	Icc (A)	Iccq (A)
Vccint	1.94356	1.000	0.950 to 1.050	0.82543	1.11813
Vccaux	2.31707	2.500	2.375 to 2.625	0.00643	0.92040
Vcco25	7.50891	2.500	2.375 to 2.625	0.04596	2.95761

Figure 2.13: Figure showing the output of the Xilinx xpower tools for power estimation of the FPGA

here are incorporated with the data from vendor specifications of each device. Such requirements are summarised in Table 2.4. It is preferable for each DAC to have its own power supplies. This is because the DAC produces large transient current requirements, determined by the output required which will be different per DAC. Therefore, an additional analogue and an additional digital 3.3v supply is added per DAC. Switch mode device utilisation is minimised in favour of linear regulators; such

Table 2.4: Table detailing the total power supply together with the target device and domain

Voltage (v)	Current (A)	Vendor	Part	Domain	Function
1.8	1.50	ON Semi	CS5253B	Digital	DAC/RAM/ENET
1.0	1.20	Linear Tech	LT1076	Digital	FPGA
1.2	0.02	Linear Tech	LD1076	Digital	Reference
3.3	0.60	On Semi	CS5253B	Digital	DAC
2.5	4.00	Linear Tech	LD1086	Digital	FPGA/ENET/RAM
5.0	1.20	Linear Tech	LTC1086	Analogue	IQMOD/BUFFER
-5.0	0.20	Linear Tech	LTC1086	Analogue	BUFFER
3.3	0.80	Linear Tech	LTC1086	Analogue	DAC
1.8	0.40	Linear Tech	LTC1086	Analogue	DAC

devices offer superior noise at high current draw at the expense of efficiency and subsequent thermal considerations [61].

Ground Plane Strategy

Correct separation of analogue and digital ground planes for optimal performance is imperative [62]. Non-optimal implementations can lead to noise and signal crossover onto either plane. For example, a large transient current drawn by the FPGA during logic operations (of up to 3A), will introduce noise to devices on the analogue plane. This may be attributed to digital ground current flow when the tracks are incorrectly routed over the planes. To avoid this common implementation problem, this system is divided into digital and analogue planes without overlapping tracks between planes [63]. Importantly, there is also an implied ground connection internal to the DAC.

There is conflicting opinion on linking ground planes at different stages [63], [64]. Therefore, this design utilises an advanced ground plane strategy, whereby the analogue and digital power supply separation is maintained throughout the design. From the mains input supply, dual AC-DC converters are fitted, which are isolated from one another. Each AC-DC supply is used as the source for the respective voltage domain regulation. The configuration is displayed diagrammatically in Figure 2.14, which also highlights incorrect track routing over domains (red line) and correct track routing with each domain (green line).

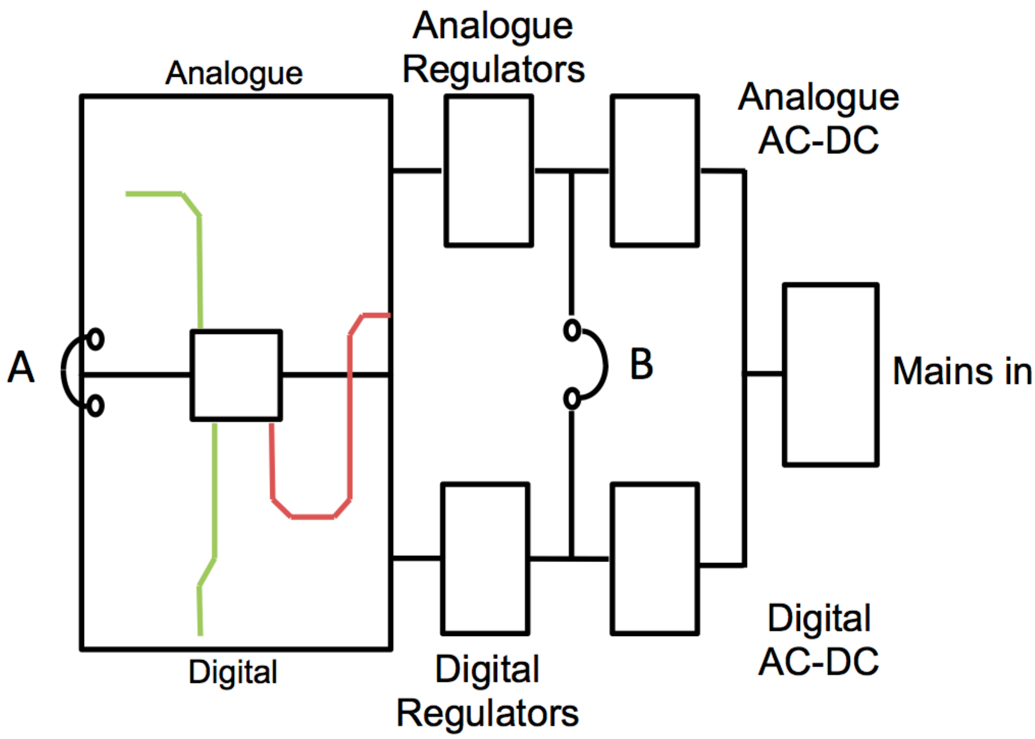

Figure 2.14: Figure showing the ground plane isolation and potential connection points: on the regulator board (A) and after the AC-DC converters (B). Also shown are correct (green) and incorrect (red) PCB track routes

In detail, the mains supply is a standard 240v 13A kettle-style plug with a fuse fitted at either end. The mains voltage is split into two live and neutral pairs, each of which is connected to a AC-DC converter. The output of the AC-DC is 9V, which is used as the input to each of the voltage domain regulator boards. The current capacity of each AC-DC is 20A, which is determined to be more than the total required current per board. Connection point B from Figure 2.14 is between the AC-DC and each of the regulator boards. Making this connection would provide a common ground plane, separated post the AC-DC converters. Connection point A is on the power PCB and making this connection would connect the analogue and digital regulator outputs together, creating a common power plane post the regulator boards. As previously detailed, the literature is not clear as to which strategy will provide the optimum result. Therefore, this strategy allows for reconfiguration as performance dictates.

2.4.7 Design Highlights

This section details general design considerations that are required as a consequence of the type and varied nature of the devices selected. Specifically, details are provided for power decoupling, the reset strategy and high frequency off-board connectors.

Power Decoupling

Power decoupling is an important consideration in complex and high frequency digital designs [65], [66]. The purpose is to provide local current storage as close as possible to the signal source. This is required to *smooth* power requirement over the PCB and eliminate current spikes during signal transition. A capacitor is used to provide such current storage, the value and quantity of which is specified in the component data-sheet. Commonality exists between the data-sheets for the placement and values of the decoupling capacitors. An example is shown in Figure 2.15 where two difference data-sheets (one for the IQ Modulator and one for PHYCeiver) show similar decoupling requirements. As a general rule, it is advisable to have a decoupling set per

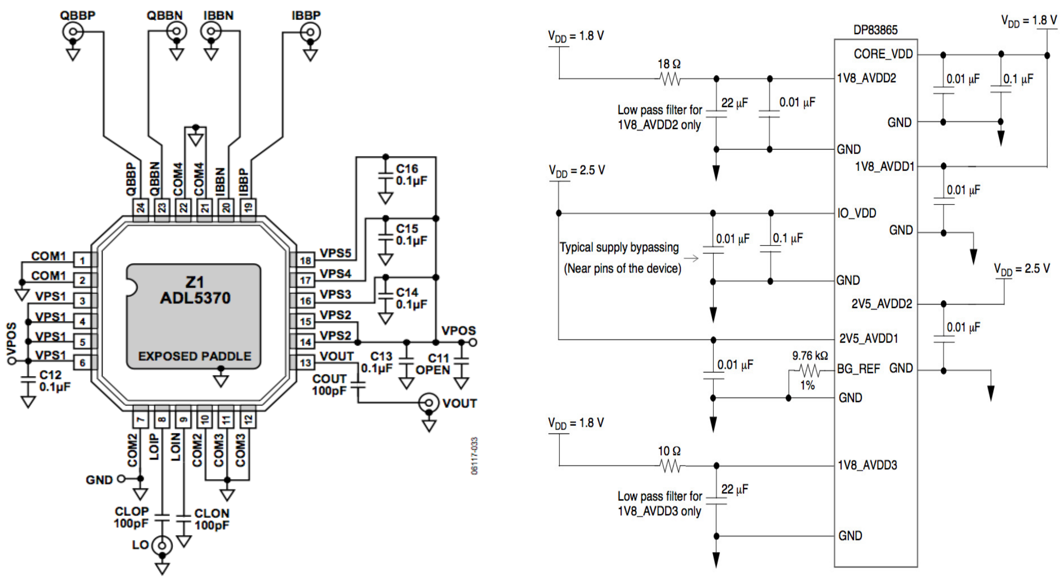

Figure 2.15: Figure displaying the similar decoupling requirements for the IQ Modulator (left) and the Ethernet PHYCeiver (right) taken from the respective datasheets

pin. A decoupling set is defined as a low value capacitor ($0.1uF$), a medium value

$(1-10uF)$ capacitor and a large value capacitor $(100uF)$. In practice this is not feasible due to size and location constraints. As an alternative, evenly distributing low and medium value capacitors (roughly equating to one low or medium value capacitor for every two signal pins and one large per 10-20 pins), satisfies the majority of decoupling requirements while maintaining a realistic capacitor footprint. Importantly, the physical location of the capacitors is still adhered to. Low capacitor values are to be within 1cm of pins, medium capacitor values within 3cm and large capacitor values within 10cm. Additionally, an attempt to evenly distribute geographically over the PCB is followed.

Reset Strategy

The devices selected necessitate both a logic high reset and a logic low reset. To accommodate the reset requirement, an on-board push switch is used which drives a single FPGA pin high. Two pins on the FPGA are subsequently driven low or high by the FPGA. The low-driven pin is connected to devices that require a logic low reset, the high-driven pin is connected to devices that require a logic high reset. This strategy is displayed diagrammatically in Figure 2.16. This strategy enables

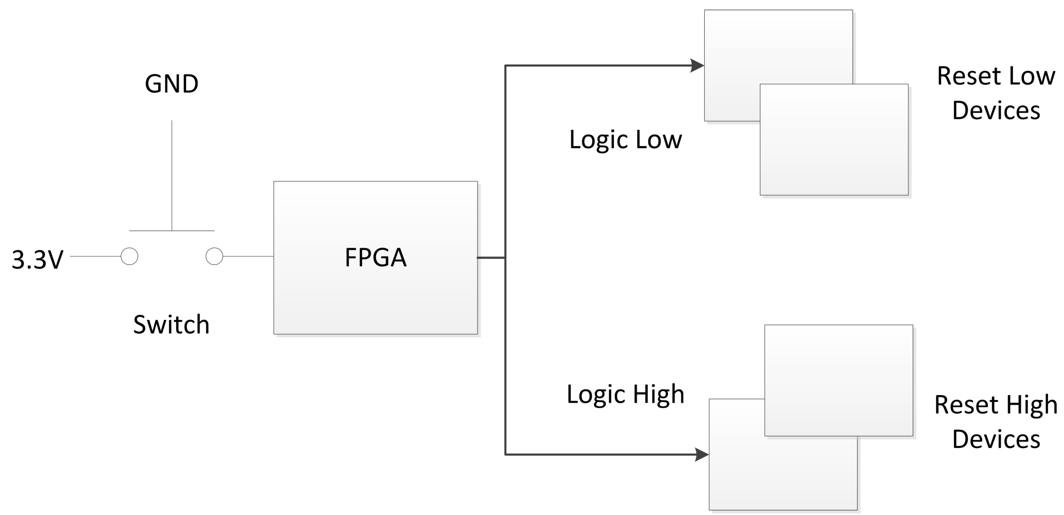

Figure 2.16: Figure showing the reset strategy employed, using the FPGA to drive logic high or low as appropriate

the FPGA to instigate a reset to any of the connected devices. Conversely, the reset button only issues a reset to the FPGA and as a consequence, the FPGA must be programmed before any of the connected devices are reset. Therefore, this method is a trade-off between reset functionality and the additional discrete electronics required to otherwise produce a logic high and logic low reset.

Off-Board Connectors

Each of the DACs has four high frequency ports. To clarify, one is for the I output, one for the Q output, one for the IQ modulated output and one for a local oscillator input. To save space, traditional PCB mounted Sub-Minature "A" (SMA) connectors are not used in favour of Sub-Minature "C" (SMC) connectors [67]. Such connectors are considerably smaller but require special cable assemblies to carry the signal to its destination. In this system, the routing from the SMC will be to a front panel containing traditional SMA connectors, so customised cable requirements are not relevant; the cable routing is detailed in later sections.

2.5 Schematic Design

The schematic design process requires each signal pin of system components to be connected together in the required manner. An example, which illustrates the complexity of such an undertaking is provided in Figure 2.17, while the complete schematic is provided in Appendix B. It is common to split a schematic in such a manner to improve readability. The figure shows the pin-to-pin interconnects for two of the DACs. In addition, the connections between each DAC and the corresponding IQ modulator is displayed.

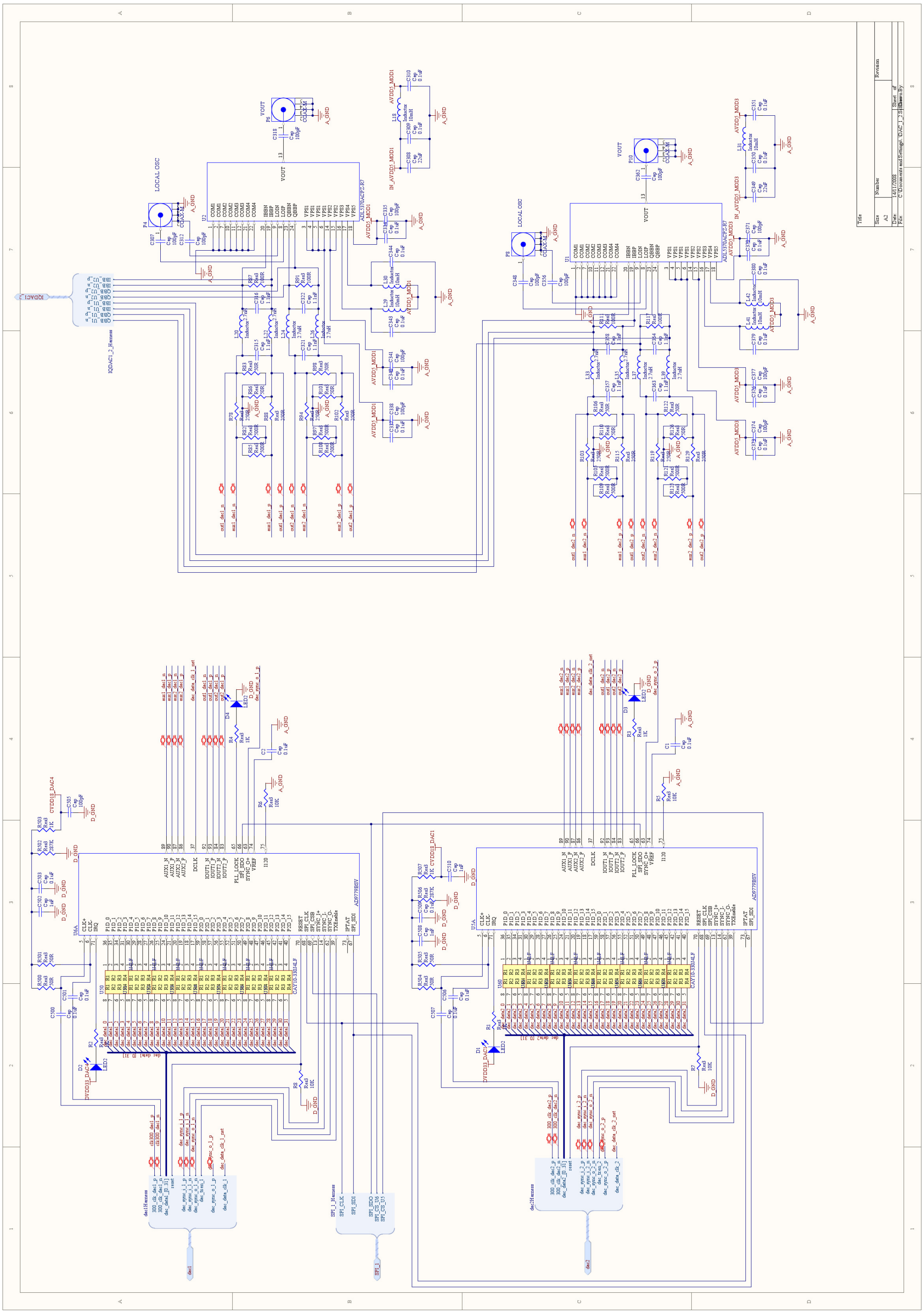

Figure 2.17: Figure showing the schematic for DACs 1 and 2, together with IQ modulators 1 and 2

During the schematic design, all previously referenced component data-sheets are consulted in detail, to determine how the devices are connected together. Diligent attention to detail is required to ensure the correct connections are made, as the schematic is used as the connection blueprint for the subsequent PCB. Additionally, each component has external component requirements such as filters and configuration resistors which are also taken into account. As each component utilises standard interconnects, such as Low Voltage Transistor-Transistor Logic (LVTTL), Low Voltage Complimentary Metal Oxide Semiconductor (LVCMOS) or Low Voltage Differential Signalling (LVDS), the interconnection function and voltage requirement is priority. It is noteworthy that the FPGA supports any of the previously mentioned interconnect standards which simplifies component interconnectivity.

2.6 PCB Design

An analysis of the device requirements leads to a PCB design strategy. Due to the system complexity and number of components it is decided to split the design into two separate PCBs; one containing the power regulators and one with all other components. This strategy enables simpler PCB routing and enables each board to be tested in isolation. The latter being important due to the high cost of the components. Although the author has previous experience of PCB routing of highly complex systems, design guides are additionally consulted [68], [69], [70], which results in the following routing and placement rules:

1. Minimum signal track width to be 0.5mm
2. Reduce (where possible) tracks between pins
3. Gap between track and device to be 2.5mm
4. Use only through-hole vias not blind vias †
5. Maximum of 45^o track bend - tracks should turn corners in three stages

† Although blind vias make PCB routing vastly simpler compared to through-hold vias, the cost of manufacture of boards with blind vias is around 10x higher

6. Minimum power track width to be 2.5mm
7. Ground planes should not be grouped together to eliminate board warping
8. Differential path lengths should be within 10% of each other
9. Clock track length should be minimised and have a width of 2.5mm
10. Analogue and digital signal tracks should not cross domains
11. Isolation of digital and analogue domains to be maintained.

Such rules are programmed into the PCB design tools and are enforced through an iterative design process. These rules, combined with the schematic, enable routing of the design to commence. A systematic PCB design methodology is utilised. First, device interconnects are analysed to ascertain a suitable device topology for routing. This includes only the major components (such as the FPGA, RAM, DACs and Ethernet). Other components are not placed yet as they would interfere with routing. Second, each of the interconnecting tracks are routed using an area wide enough to have all tracks routed together. Critical signals are routed first (such as clocks, or high speed interconnects) followed by control signals and finally debug and JTAG signals. As the routing continues and the PCB becomes more crowded, more layers are added to maintain the routing discipline. During the routing process, it is important to be mindful of additional components that are not yet placed (such as decoupling capacitors) and allow appropriate space. The third stage is to add in the remaining components and review the top and bottom silkscreen. This layer provides a visual pointer to each component location, polarity and identifier (name, number etc.). If the silkscreen is incorrect or not clear it may hinder the build or debugging process. An illustration of the completed PCB layout with all layers visible, is provided in Figure 2.18. The layout for each layer of the design can be found in Appendix C. Development time for the schematic was around 4 weeks, with an additional 6 weeks for the PCB. Subsequently, files that describe the PCB per layer are sent out for manufacture (known as *Gerber* data). This process takes an additional 2 weeks. The main PCB consists of 24 layers in total and the power PCB consists of 6 layers in

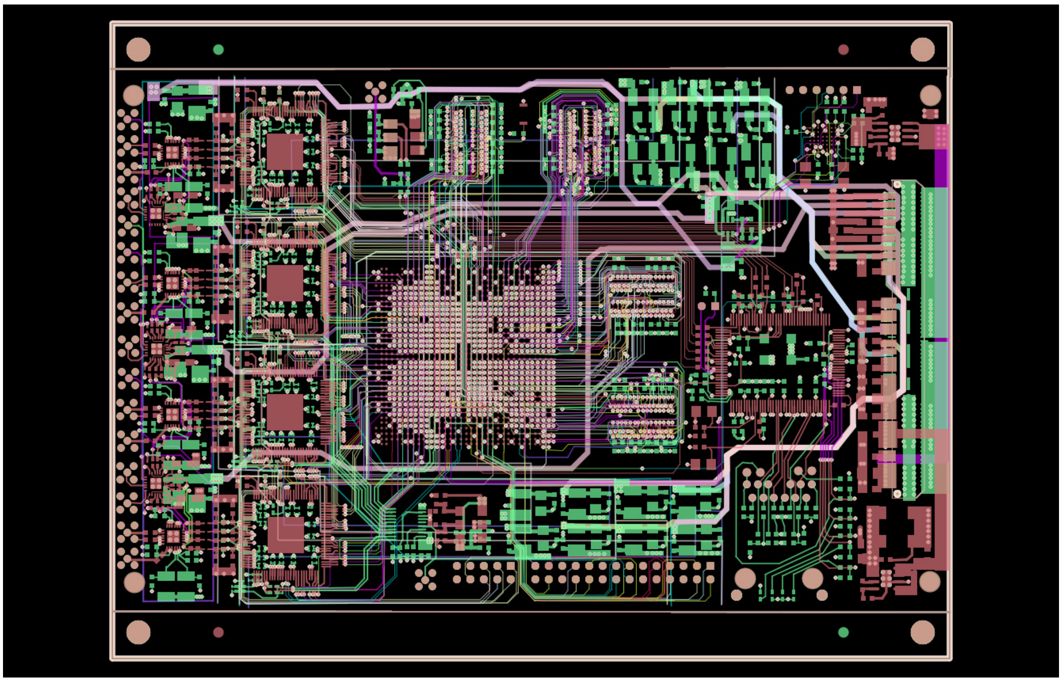

Figure 2.18: Figure illustrating the PCB design with all layers visible

total. The power PCB is illustrated in Figure 2.19. Unfortunately, the main PCB was not photographed before components were fitted.

2.7 Build

With the PCB complete, the required parts are procured. The component list (so called Bill Of Materials (BOM)), contains components for many manufactures, vendors and distributors. There are over 2500 individual components required for the system. Therefore, a component distributor or kitting house is used to supply all components which simplifies this process considerably. The components, the PCB and the BOM are combined together. This completes the requirements for manufacturing to commence (termed as a *kit*). Board manufacture is performed by a sub-contracting specialist. All required components are fitted using surface-mount machinery according to the BOM, together with integrity checking using optical methods.

During the build process, it is noticed that the footprint of the Ethernet chip is incorrect. Specifically, the position of pin 1 is mirrored and the order of the pins runs

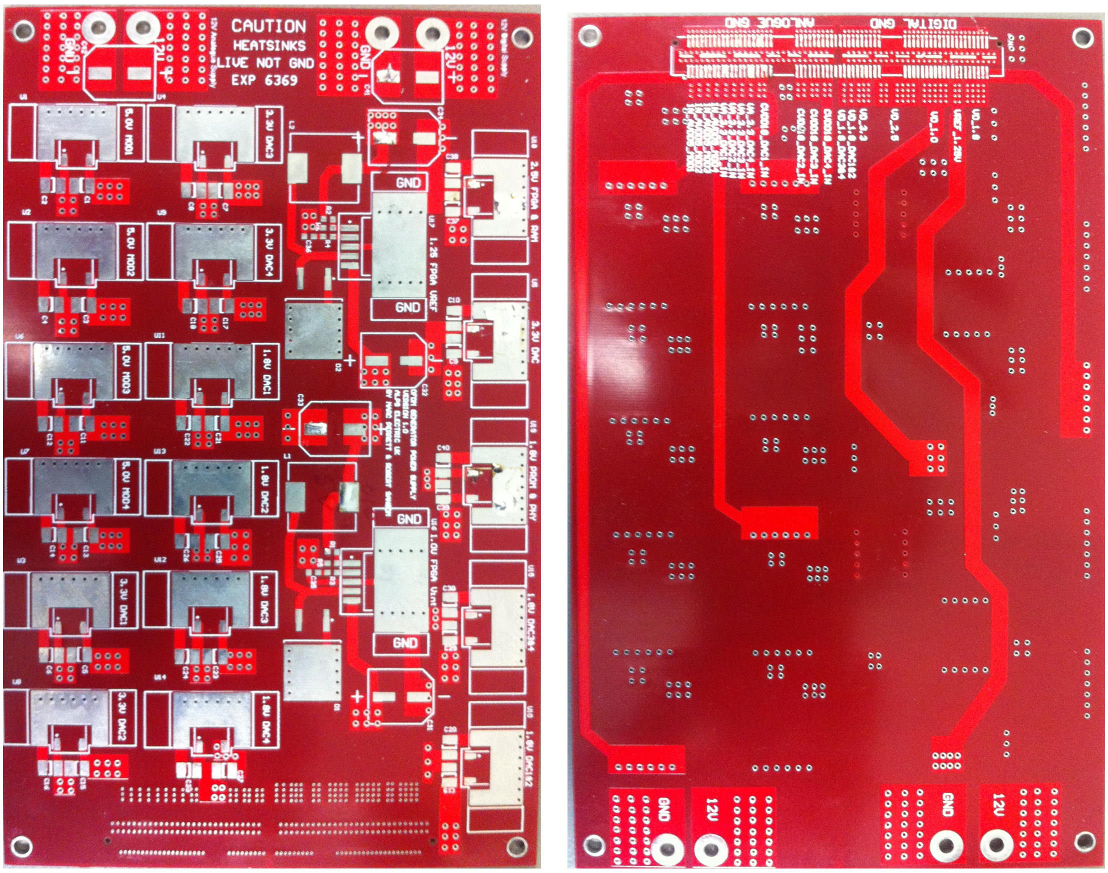

Figure 2.19: Figure displaying the unpopulated power PCB top (left) and bottom (right)

counter-clockwise instead of clockwise. This is because the device data-sheet details the pin layout viewed from below and not above (as is more common), which was not noticed during the design stage. The remedial action is to produce an additional PCB, which contains the correct pin configuration on the top layer and the incorrect (matching the main PCB) on the bottom layer. This board can be seen in Figure 2.20. This PCB is fitted on the board using the same equipment detailed earlier. Once the placement of the small PCB is complete the Ethernet device is placed over it. This completes the rework, which is illustrated in Figure 2.21. The completed main PCB is displayed in Figure 2.22. The FPGA is situated in the centre surrounded by the RAM devices. To the far right is the Ethernet PHYCeiver and Ethernet connector, while the far left shows the DAC devices, the IQ modulators and the high frequency analogue connectors.

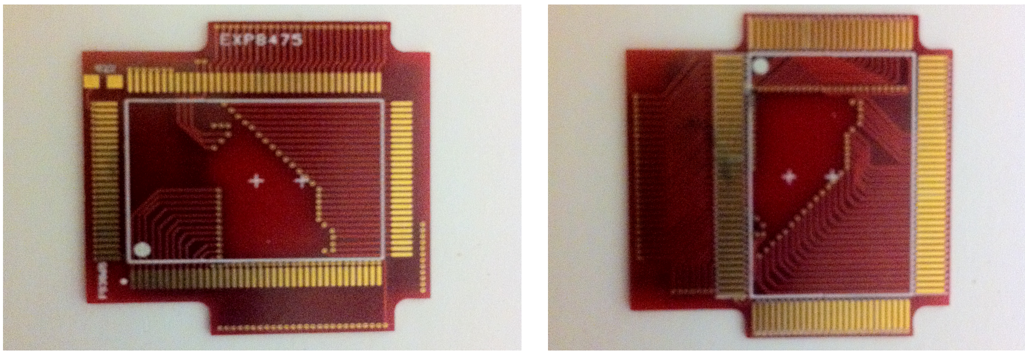

Figure 2.20: Figure showing the top (left) and bottom (right) of the rework PCB. The top has the component soldered to it and the bottom is soldered to the main PCB

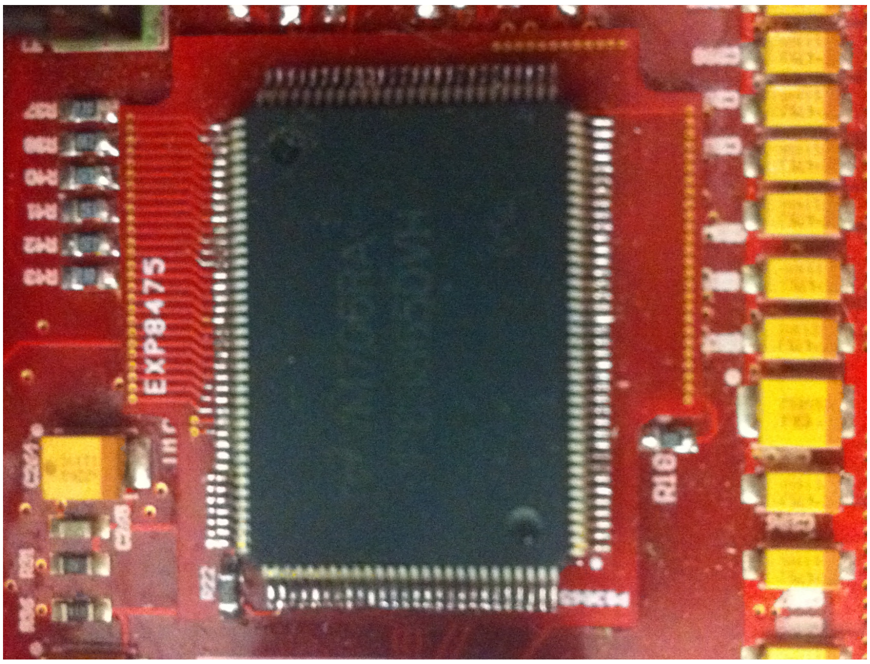

Figure 2.21: Figure illustrating the Ethernet device fitted to the pin translation PCB due to an incorrect main PCB footprint

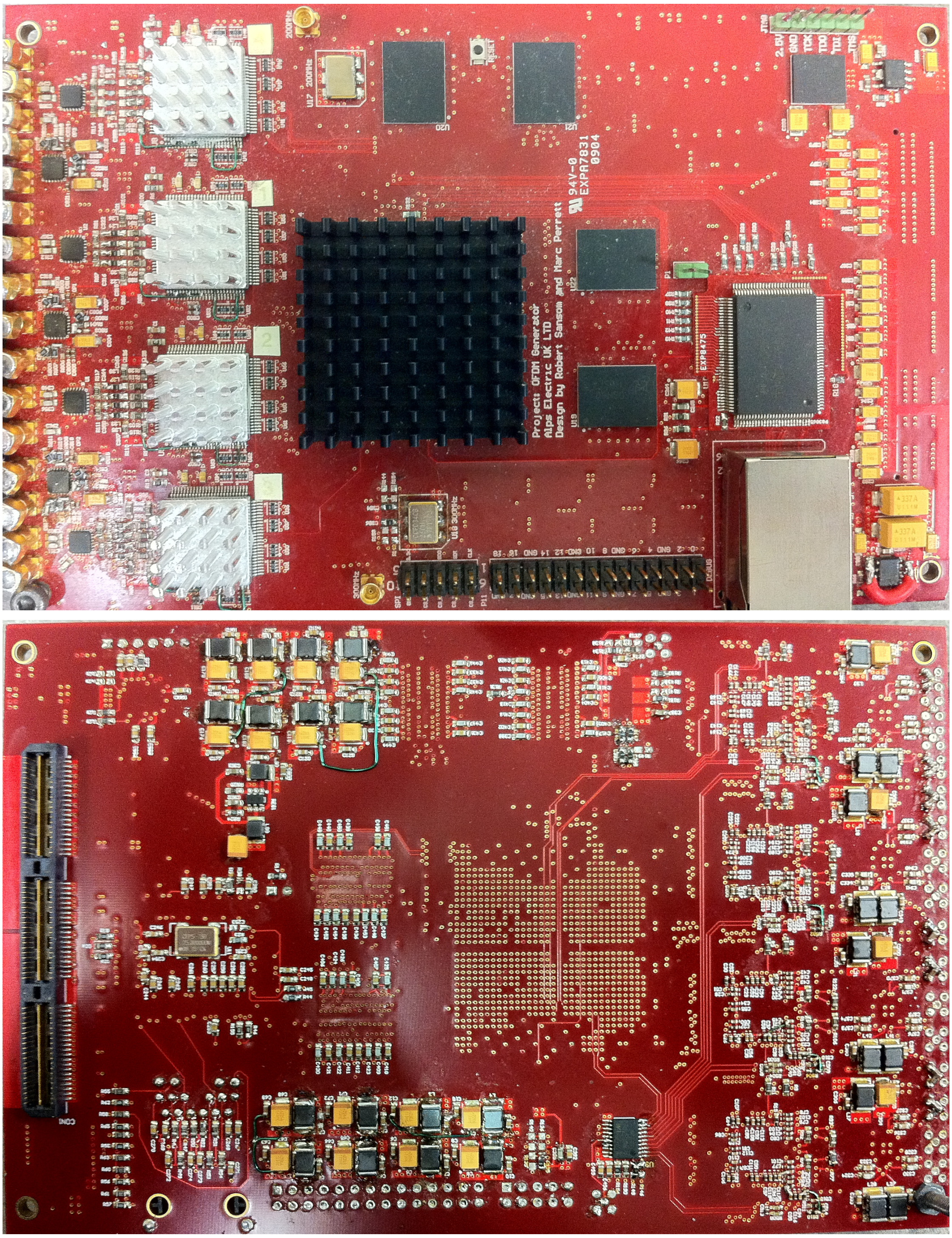

Figure 2.22: Figure illustrating main PCB top layer (top) and bottom layer (bottom) populated with components after manufacture

The completed power board is shown in Figure 2.23. Only the top is illustrated, showing 19 regulators masked by their respective heat sinks. The bottom of the power board does not have any components fitted (see Figure 2.19).

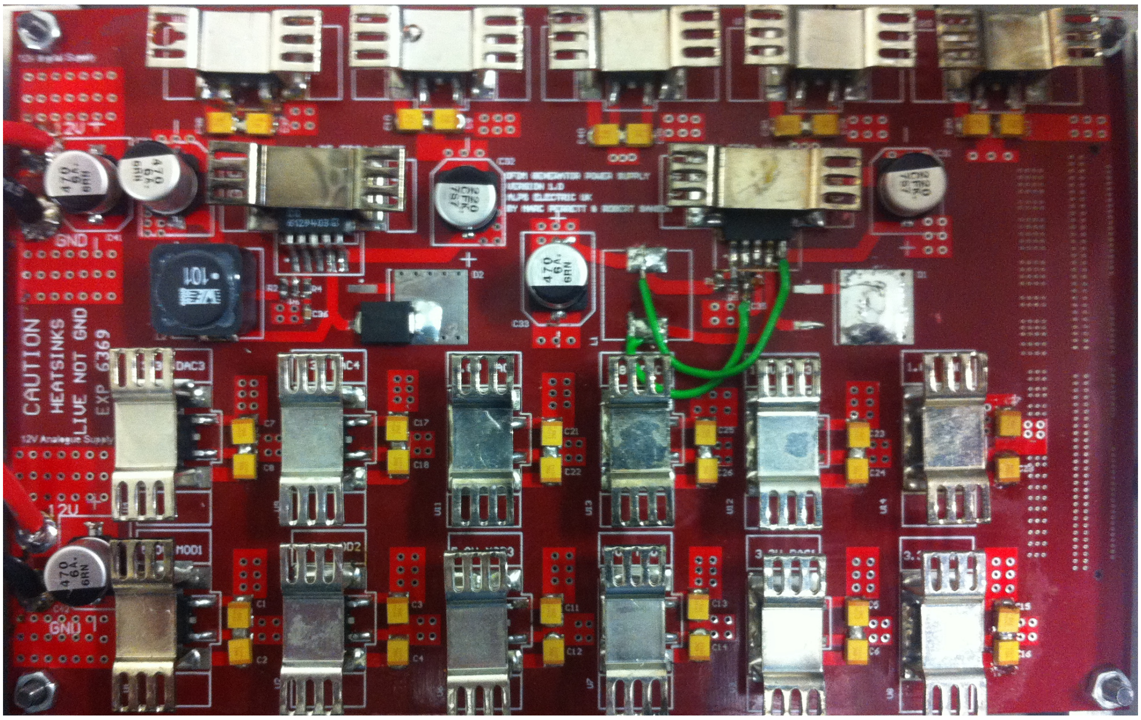

Figure 2.23: Figure showing the power board populated with components after manufacture

2.7.1 Enclosure

To provide an environment for the system during use, an enclosure is selected to house the PCBs. The design of this is two-fold - The PCB has mounting plates attached to secure it inside the enclosure and the enclosure itself has gaps drilled for access to external interfaces. Both are illustrated in Figure 2.24. The enclosure allows access to the PCB Ethernet connector, the Standard Peripheral Interconnect (SPI) USB connector, power input, debug output and JTAG. Additionally, the front panel houses four SMA connectors per DAC. Each DAC has an I output, a Q output, an IQ modulated output and a local oscillator (for the IQ modulator) input. One cable per SMA is routed from the main PCB to the front panel. The cable routing and the front panel connectors are illustrated in Figure 2.25. The final configuration with the

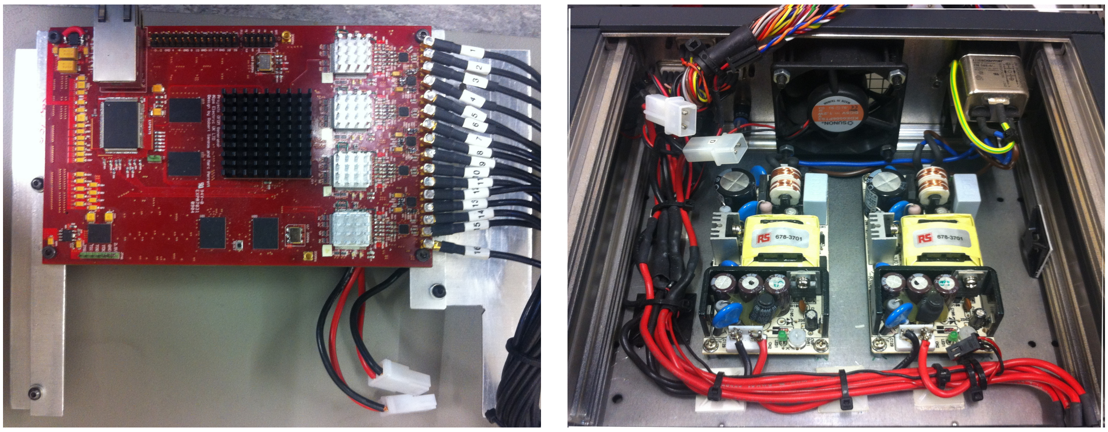

Figure 2.24: Figure displaying the main and power PCB with mounting plates fitted (right) and the enclosure before the PCBs are inserted (Left)

system housed in the enclosure is shown in Figure 2.26. In this example, the front and top panels of the enclosure, and the front panel cables has been removed, so that the internals of the enclosure can be viewed.

2.8 System Bring-up and Test

The system contains a combination of high value analogue and digital devices. Therefore, it is not recommended to transition from build to test and verification in a single step. Instead, a systematic process, known as *bring-up*, where individual block verification is employed so as to minimise the chance of damage to any single component. Principally, correct operation of the power supply must first be confirmed which is detailed in Section 2.8.1. Analysis of the voltage output and additional factors such as noise and current capacity are tested for adherence to specification. A *shell code* is then downloaded to the FPGA, as detailed in Section 2.8.2. This is defined as Very high speed integrated circuit Hardware Description Language (VHDL) code which consists only of correct pin mapping between connected devices, together with code to verify the core operation of the such connected devices. Initially, the basic operation of the FPGA is confirmed before the shell code is extended for memory testing (Section 2.8.3), Ethernet (Section 2.8.4) and DACs (Section 2.8.5).

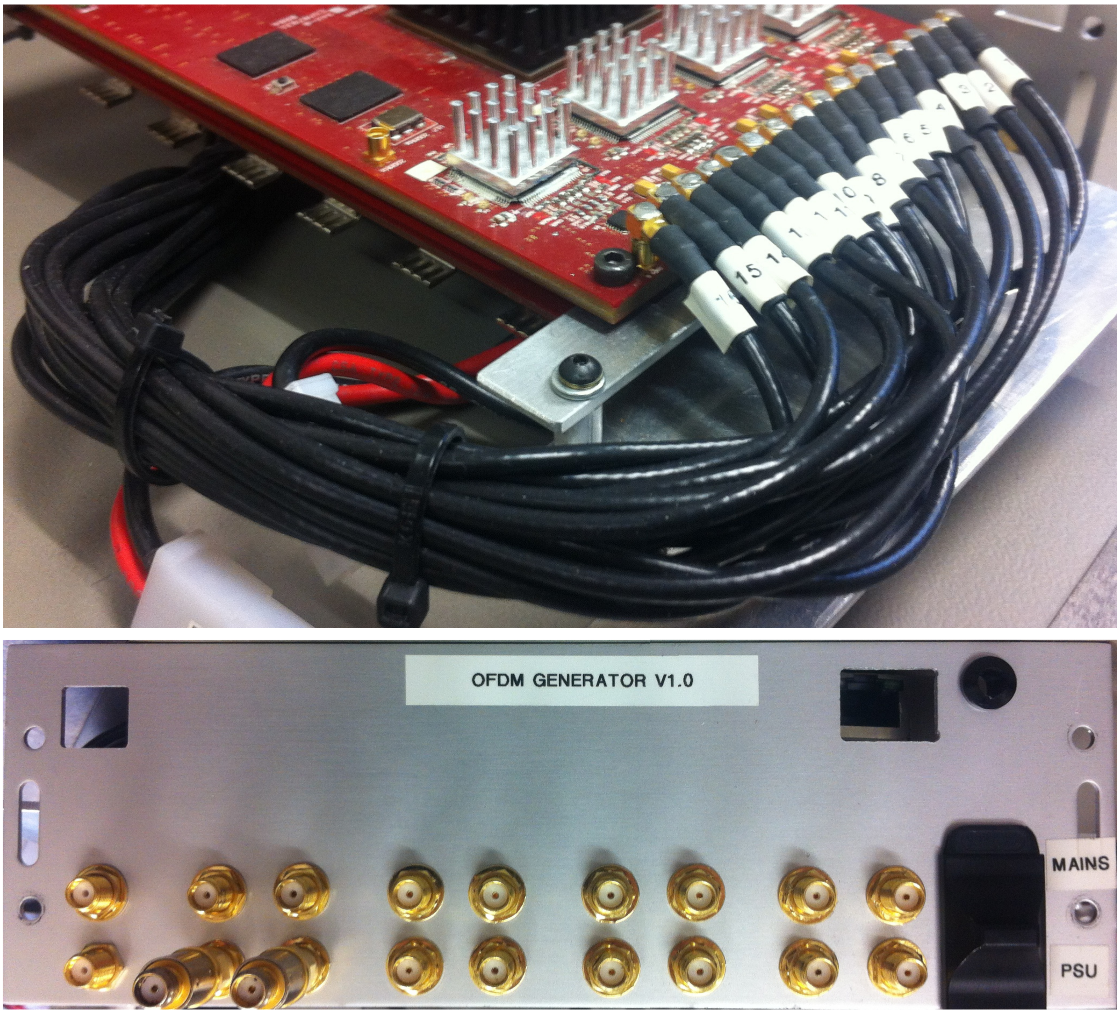

Figure 2.25: Figure illustrating each cable from the main PCB (top) and the front panel of the enclosure with SMA connectors (bottom)

2.8.1 Power Supply Testing

The power supplies are situated on a separate PCB and therefore can be evaluated completely independently of the main system. The board is connected to a bench power supply and a Digital Multi Meter (DMM) is used to measure the output voltages. Such a measurement is taken with an appropriate load placed at the output of each regulator using the pre-determined current requirements detailed in Table 2.4. The results of the measurement can be found in 2.5. The results indicated that all power supplies are functioning correctly, with the exception of the 1.0V supply which is reporting only 0.12V.

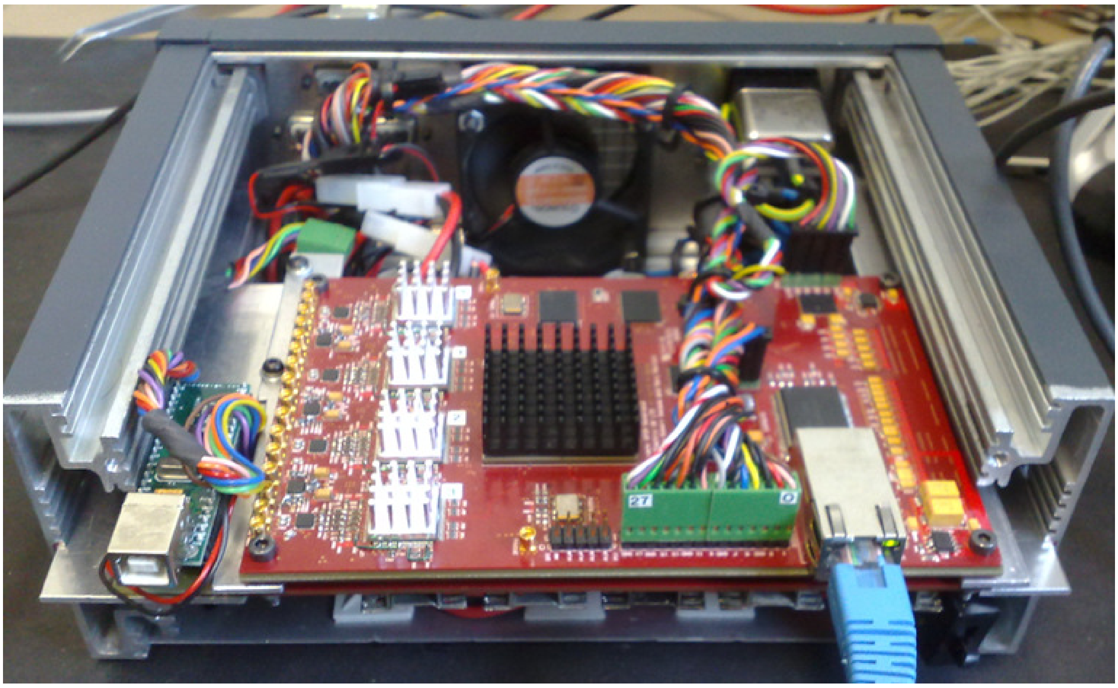

Figure 2.26: Figure showing the final configuration of the two PCBs and the enclosure

Table 2.5: Table detailing the initial power supply results per voltage under required current load

Predicted (v)	Actual (v)
Digital 2.5	2.47
Digital 1.2	1.22
Digital 1.0	0.12
DAC 1 2.5	2.56
DAC 2 2.5	2.54
DAC 3 2.5	2.51
DAC 4 2.5	2.53
DAC 1 1.8	1.77
DAC 2 1.8	1.79
DAC 3 1.8	1.77
DAC 4 1.8	1.80
DAC 1 3.3	3.31
DAC 2 3.3	3.33
DAC 3 3.3	3.31
DAC 4 3.3	3.30
IQMOD +5	5.07
IQMOD -5	-4.89

1.0V Supply Investigation

The current capacity of the LT1076 device is insufficient at the required voltage, which was an oversight during the design phase. A search is undertaken to identify an appropriate replacement part that can provide the required current and fit within the available pin layout. No device is found that meets both requirements and instead, a part is selected that can provide the required current and will integrate into the available pin layout with minor PCB modifications. Additionally, the part chosen (MIC49300) from Micrel [71] is not of a switch mode type, but is instead a linear rectifier as a suitable switch mode part could not be identified. This change is accounted for in the remedial action taken. Table 2.6 details the pin mismatch between the devices. The feedback and adjust pin (pin 1), the Vsw and Vin pin (pin 4) and

Table 2.6: Table detailing the pin mismatching between the original 1.0v power device (LT1076) and the replacement part (MIC49300)

Pin No.	**LT1076**	**MIC49300**
1	Feedback	Adjust
2	Vc	Vbias
3	GND	GND
4	Vsw	Vin
5	Vin	Vout

the GND pin (pin 3) are equivalent in operation and require the same external configuration circuitry. Therefore, no modification is required for these pins. However, pin 2 (Vc) of the LT1076 is equivalent to pin 5 (Vout) of the MIC49300. Furthermore, pin 2 (Vbias) of the MIC49300 needs to be supplied with source voltage. Therefore, the track that connects the original output (pin 2) to the rest of the PCB is cut and instead wired to a nearby input voltage pad. The pin on the device destined for the original input pad (pin 5) is lifted from the board (i.e not soldered to the original pad) and routed, using a wire, to a nearby output pad. All other pin connections are unchanged and the result of this modification is illustrated in Figure 2.27. The voltage and current measurements are repeated for the 1.0V supply, the result of which can be found in Table 2.7. Finally, noise assessment of each of the power supplies

Table 2.7: Table detailing the 1.0 power supply results after remedial modification of the PCB and the change of device (under current load)

Predicted (v)	**Actual (v)**
Digital 1.0	0.97

output was carried out using a digital oscilloscope. In all cases the noise levels were

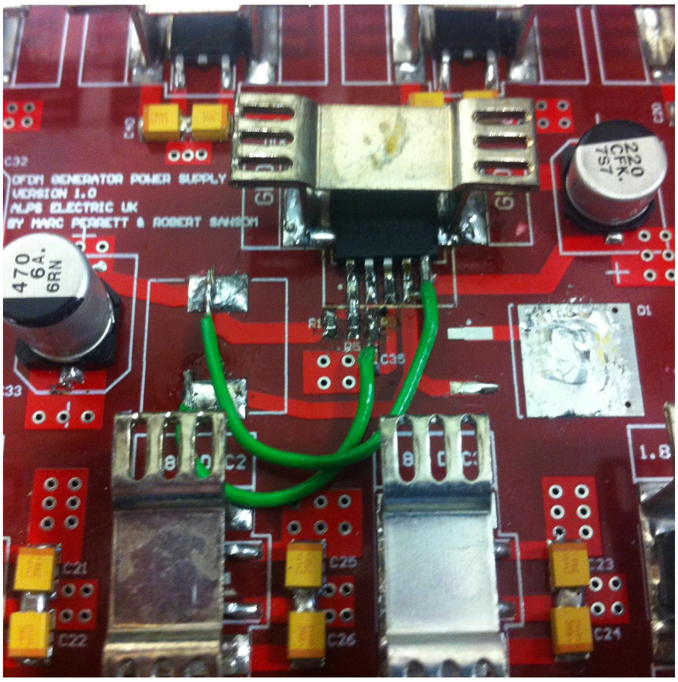

Figure 2.27: Figure showing the modification to the digital 1.0v power supply to rectify the incorrect layout.

below 1% with the ground planes connected at point A (from Section 2.4.6). When point B was used the noise was significantly increased. Altogether, the results of all measurements are satisfactory and the power board can be used to provide power for the main PCB for further testing.

2.8.2 FPGA Test Using Shell Code

With power provided to all parts of the system, individual components can be tested for correct operation. As all devices are connected to the FPGA, it is not possible to test individual components in isolation. Therefore, the FPGA is programmed with a code that allows for basic testing of connected devices, termed as the *shell code*. A shell code maintains the correct pin allocation and minimum requirements for external devices to operate correctly. This includes Ethernet MAC connectivity (required for the PHYCeiver chip to complete speed auto-negotiation), DDR memory interface and DAC clocking requirements. Importantly, such code can be extended to perform a more detailed investigation into the connected devices as required. Before such code can be used, the FPGA is tested for connectivity and the on-board flashing capability.

Xilinx provides an FPGA programming solution (called a platform USB, which converts a USB interface (user facing) to a JTAG interface (FPGA facing) [72]. JTAG is a communication interface standard that allows access to the programmable portion of the FPGA. Additionally, devices can be daisy-chained together and programmed sequentially. Such a configuration exists between the FPGA and the on-board flash used for program storage. Therefore, the FPGA can be programmed in two ways: directly over JTAG which is erased upon power cycling, or indirectly by programming the flash which subsequently programmes the FPGA upon a power cycle. The JTAG connection configuration is illustrated in Figure 2.28.

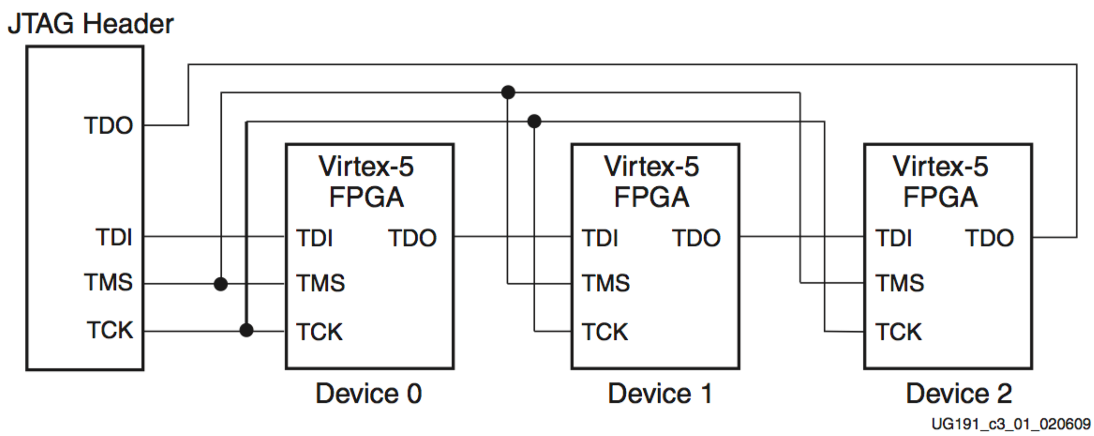

Figure 2.28: Figure showing the FPGA JTAG configuration as a daisy-chain of devices to a common header connection (taken from vendor datasheet)

JTAG Connectivity Test

The vendor tool (called IMPACT) is loaded and the direct programming of the FPGA initiated. The tool reports if the programming is successful or not. Furthermore, by monitoring the power draw of the system, it is possible to determine the success if an increase in current demand is observed. The file is programmed and the process is deemed successful by the tools. A significant current draw is also observed after programming, indicating that the FPGA is successfully programmed and is also driving external components.

Flash Programming Test

The file used to test the direct programming of the FPGA is re-targeted to a format appropriate for the on-board flash. This is completed using the vendor tool, by selecting the required flash part and source file. The flash is programmed using the same JTAG interface previously outlined. The power is then cycled and monitored to see if a similar current increase is observed. The same current draw is observed as before upon power up. Therefore, the flash programming is proven successful and in future, a design that is loaded can persist in the system without the need to reprogram with every power cycle.

2.8.3 Memory Testing

The shell code, introduced in Section 2.8.2, is extended to enable testing of the DDR memory. A Linear Feedback Shift Register (LFSR) is used to generate a Pseudo Random Binary Sequence (PRBS) data set, which is sufficiently large to provide a non-repeated output for the entire RAM. Such a dataset is written into the RAM by the FPGA, starting at the first location and completing at the last location. The RAM is subsequently read in the same order and each value compared with the expected PRBS value. If the check is completed successfully a debug pin is asserted high, if the check is unsuccessful the pin stays low; such a pin exists for each of the four RAMs. It is observed that all four pins drive high which demonstrates the RAM devices are functioning correctly. Therefore, the non-standard memory interface detailed in Section 2.4.4, together with the other components concerned are correct and working as required. Furthermore, as the test exercises all writeable and readable areas of the memory it guarantees the integrity of subsequent data stored using them.

2.8.4 Ethernet Testing

The Ethernet interface cannot be tested in isolation and instead is evaluated with the FPGA. Therefore, the shell code is extended to include the Xilinx-supplied MAC and connected to the pin locations which connect to the PHYCeiver. The MAC RX interface is output to system debug pins which are subsequently connected to a Logic Analyser. In this configuration, any traffic received over Ethernet can be observed. The aim is to determine if there is a communication pathway between the Host PC, the on-board Ethernet interface and the MAC inside the FPGA. The test procedure is performed by connected the magnetic interface to a Host computer, at which point an Address Resolution Protocol (ARP) transaction will be instigated. The ARP packet will be transmitted over the Ethernet interface, through the PHYCeiver to the MAC inside the FPGA and then to the Logic Analyser. An explanation of the function of ARP and low-level details of the Ethernet interface are provided in Chapter 4. Additionally, there are Light Emitting Diodes (LEDs) on the board, that reflect the current Ethernet link status. Indication of the link speed (10/100/1000), duplex mode (half or full) and link status (connected or not connected) are available. When the Ethernet cable is connected, these status LEDs are monitored to ensure the status reported matches the status configured in the Host computer. For this test, the Host is configured to full-duplex at 100TX-speed and the correct LED status is observed. The link speed and status are changed on the host and the LED status changes accordingly. Furthermore, the Logic Analyser is interrogated to ascertain if the Host transmitted ARP packet is received successfully. Such a transaction is observed and is deemed correct. This provides sufficient evidence that the Ethernet interface is connected correctly; this interface is subject to further analysis in Chapter 4.

2.8.5 DAC Testing

The DAC output voltage is determined by a value provided by the FPGA, as detailed in Section 2.4.3. Therefore, the shell code is extended to provide test values to the DAC and the resulting analogue signal measured and analysed. For this test, the output value to the DAC is transitioned from the maximum ($7FFF_H$) to the minimum

(8000_H) every clock pulse. The expected output is a square wave at a frequency of half the DAC clock. However, the observed output is very noisy and points to a serious problem with the DAC. It was found that there was a short between the analogue and digital ground planes. Analysis of the PCB layout of the DAC and the vendor data-sheet highlights a layout error. The DAC has both analogue (AGND) and digital (DGND) ground pins. The pin (CGND) is mistakenly connected to the digital ground. The pin-out of the DAC and the area under discussion is displayed in Figure 2.29. The remedial action to rectify the error is two-fold. Firstly, the pins

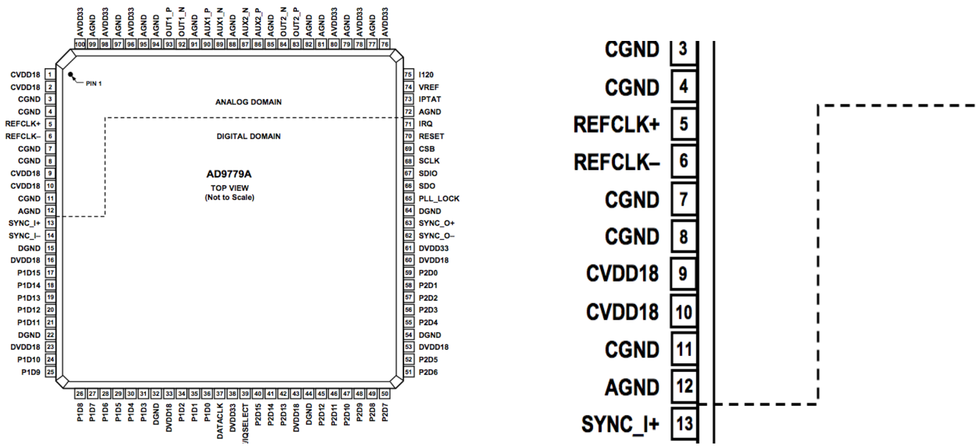

Figure 2.29: Figure illustrating the AD9779 pinout (right) and the erroneous ground pin (left) (taken from vendor datasheet)

AGND and CGND are lifted from the PCB and are connected, using track-wire, to a nearby analogue ground point. Secondly, the PCB track connected to the decoupling capacitors required for these pins is cut and reconnected to AGND. The resulting modification can be seen in Figure 2.30. The test previously applied to the DAC is repeated and the result measured as before. In this test the output is as expected. All four DACs are testing using the same method and are found to operate correctly. It is unknown if the output is to specification as each DAC requires a complete characterisation, which is performed in later chapters.

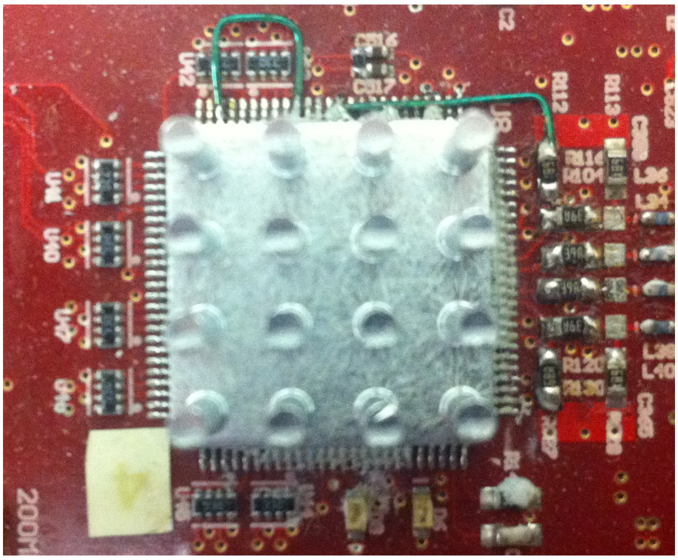

Figure 2.30: Figure to illustrate the modification to the DAC CGND and AGND pins to correct erroneous pin layout

2.9 Conclusions

This chapter describes the design, from initial specification to final deliverable, of a bespoke platform for the generation of wireless signals. Detailed is a review of the state of the art components (as available in 2007) together with justification for the design undertaking. System specifications are produced which lead to considerations of component selection and configuration. Additionally, specific areas where the design of the system requires greater effort are detailed. A detailed account of the design of the schematic and two PCBs is included, together with design guidelines required to produce an appropriately performing system. The build procedure of the system is summarised with example of how specific components are positioned and soldered to the PCB mechanically.

A systematic approach to board bring-up is detailed, starting from power requirements and then to the development of a shell code for the FPGA. Such code is used to test the basic functions of the FPGA before being extended to test other system components; namely; RAM, DACs and Ethernet. During this process, design errors are discovered and remedial modifications are proposed and implemented successfully.

The systems function is demonstrated successfully and preliminary test results suggest acceptable performance. The remedial modification on the power supplies and DAC ground is not expected to impede performance. Thus, the system detailed in this chapter meets the required use case and specification, and the system can therefore be qualified as a success. However, more detailed characterisation, specifically of the analogue output section, is required. It is accepted that such characterisation may identify lower analogue performance, due to the non-ideal behaviour of the buffer amplifier and the configuration of the circuity at the output of the DAC. A detailed study of the FPGA characterisation and the analogue output signals is therefore the subject of the following chapter.

Chapter 3

Characterisation of an FPGA-Based Signal Generator

Publications relevant to this chapter are detailed as follows:

Perrett, M. and Darwazeh, I. *Characterisation and verification of an FPGA signal generator for spectrally efficient wireless FDM*, IEEE International Conference on Telecommunications (ICT'2012), Beirut, Lebanon

The previous chapter detailed the design, from conception to final deliverable, of a Field Programmable Gate Array (FPGA) based wireless signal generator. Much of the functionality of the system has been tested and verified as correct to specification or, in the case where a deviation from specification is identified, remedial action has been performed and the specification updated accordingly. Furthermore, where component interconnection exists verification is performed to confirm correct intercommunication. However, the end to end performance (from digital data which is input using the Ethernet interface, to an analogue signal output using the Digital to Analogue Converters (DACs)) requires characterisation before meaningful measurements can be made to subsequently generated signals *.

*Comparisons to commercially available hardware with similar functionalities to the system under discussion, with the exception of FPGA processing, is not detailed in the thesis. In 2008-2009

The system introduced in Chapter 2 is subjected to stimuli to ascertain characterisation performance [52]. As the system contains a mixture of analogue and digital elements, often defined as *mixed signal*, a holistic approach is taken. Here, the digital element of the design is used to generate stimuli for the analogue element. In this configuration the entire system is included and an overall measure of performance can be achieved.

The following sections detail the characterisation that is performed on the system; namely transient measurements (Section 3.1), frequency measurements (Section 3.2), noise and distortion measurements (Section 3.3) and amplitude linearity measurements (Section 3.4). These results qualify the system performance and enable a demonstration, in Chapters 5 and 6, of this system to generate non-standard wireless signals. Such methods are exemplified in [73], based on earlier work in [74]. The work is concluded in Section 3.5.

The methodology to generate stimuli for measurement is by the generation of appropriate signals internally (within the FPGA), which are then passed through the DAC and associated electronic circuitry. Performance is assessed by contrasting these measurements against those based on signals generated from a reference source. In order to make data comparison and processing as simple as possible, post-processing and comparison of such signals is completed in Matlab. This presents an additional requirement of an Analogue To Digital Converter (ADC) [75], to recover the post-DAC analogue signal. To maintain the integrity of the measurements, industry standard equipment is used to sample at very high speed and pass the resulting digitised signal to Matlab as an ASCII file.

To ensure the capture process does not impact the measurements, signals generated

there were commercially available Arbitrary Waveform Generators (AWGs), however, such AWGs bandwidth was limited, in comparison to the system designed in this thesis (125MHz bandwidth). Leading examples are: Agilent 33522A (30MHz bandwidth), National Instruments NI 5422 (50MHz bandwidth) and Rhode and Schwarz AM300 (50MHz bandwidth).

from an equivalent reference source are captured in the same manner and used for calibration. Additionally, measurements gathered from the capture process are confirmed with a functionally equivalent measurement instrument, for example a spectrum analyser for frequency measurement. This methodology is summarised diagrammatically in Figure 3.1.

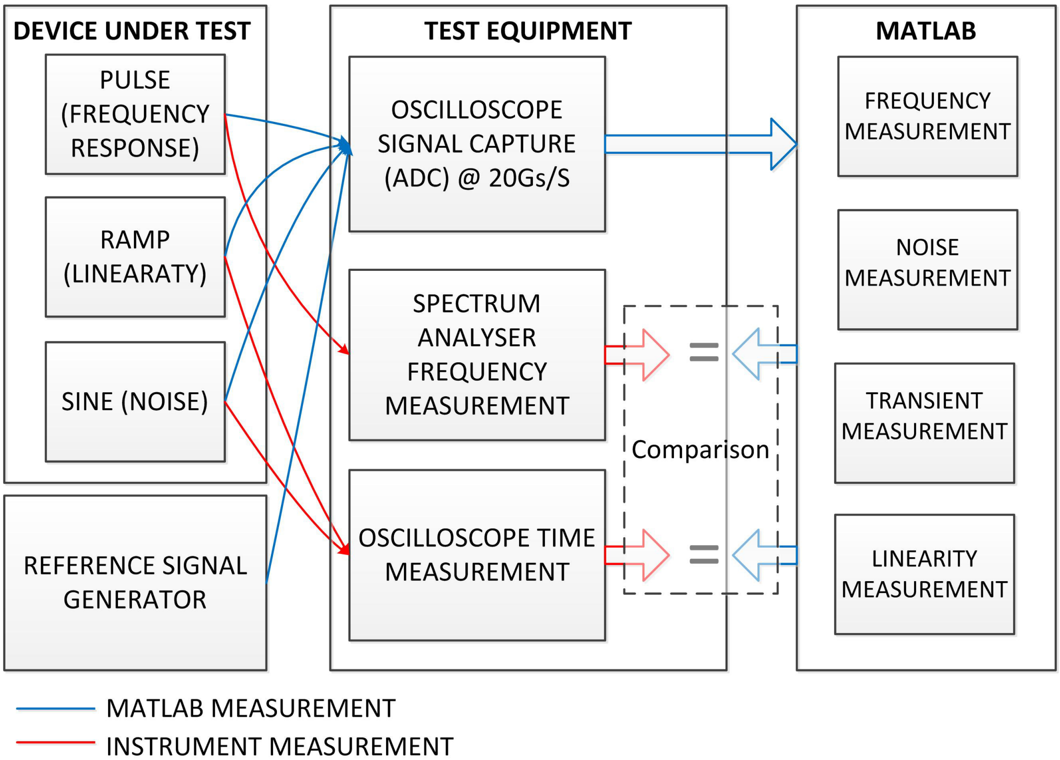

Figure 3.1: Characterisation methodology and apparatus configuration

In detail, the device under test references the system detailed in Chapter 2. The system FPGA is used to generate stimuli digitally, which is output to the analogue part of the system. The digital signal is assumed to be as close as possible to a perfect signal representation (within the bounds of the bit-depth used to represent such signals). The output of the analogue portion of the system is compared to the original digital input; the difference (i.e the error) defines the system performance.

The error is quantified by capturing the resulting analogue output of the system using industry standard test equipment. The equipment used is a Lecroy Wavemaster 9300 real time oscilloscope, a Hewlett Packard Signal generator and a Hewlett Packard 8930 Spectrum analyser. Such equipment is calibrated to a specification higher than the signal under analysis. Therefore, the signal measured will represent as close as possible the true system characteristics. Standard test equipment is generally cumbersome when synthetic measurements, such as Signal to Noise Ratio (SNR), are attempted. To aid post signal analysis, the output of the test equipment is instead passed to Matlab for processing.

With the data in its raw (i.e digitised) format, performing measurements on the data is trivial and can be highly automated using Matlab. The results of the Matlab processed data is compared to that measured using the test equipment. This is to confirm that the Matlab processing is correct. For example, frequency curves can be quantified and compared between Matlab and test equipment. Any difference would indicate an error in the Matlab processing.

To provide a reference for all of the measurements and to ensure that the test equipment or Matlab processing is not introducing errors, a reference signal is captured in parallel. Such a signal is subject to the same procedure as signals generated using the FPGA and therefore qualifies the methodology. If the performance of the reference signal is lower than expected (by comparison to the specification of the reference source equipment) it can be assumed that the capturing process is causing errors.

3.1 Transient Measurements

Transient measurements identify the time-domain properties of a systems response to a pulse or square wave stimulus. The FPGA is used to generate a single narrow pulse at the highest possible frequency. This is the narrowest possible pulse width and therefore has the widest frequency spectrum and the fastest potential zero to maximum voltage transition.

To generate such a pulse, data is transitioned from zero to the maximum possible value on consecutive clock edges of the DAC clock. Data is represented by 16-bit words, equating to $\pm 2^{15} - 1$ possible values; the maximum positive value is 32767. This data is processed by the DAC and the resulting analogue signal is captured using a high sampling rate oscilloscope. The data is then passed to Matlab for off-line processing. The post-processed Matlab signal is compared to the original oscilloscope capture to ensure no error exists in the capturing process.

Due to the high sampling rate used to capture the analogue data which creates large data-sets per capture, the total sampling period is limited to a single narrow pulse-width. This results in $\frac{10*10^9}{1/20*10^9} = 200$ data points. The Rise Time (the time between 10% of the signal and 90% of the signal) and Peak Time (the time between 1% and 100%) are measured in Matlab.

A filter is applied in Matlab to enable analysis of the signal only within the required bandwidth between DC and the DAC clock frequency. This is required as the oscilloscope has a bandwidth much greater than required. In order to ensure that this filter does not distort the measurement both the filtered and unfiltered responses are plotted and visually examined. Figure 3.2 illustrates each DAC I and Q response. Each plot contains the unfiltered (red) and filtered (blue) response. Peak Time can be used to provide an estimate of the frequency response of the system. If the system had infinite bandwidth, the Peak Time would be zero; a Peak Time of greater than zero suggests some bandwidth limitation of the system. Table 3.1 summarises the Peak Time measurements of each DAC I and Q output. It can be seen from Table

Table 3.1: DAC 1 Transient Results

DAC	**Metric**	**DAC I**	**DAC Q**
1	Rise-Time	2.55ns	2.55ns
1	Peak-Time	5.40ns	5.45ns
2	Rise-Time	2.60ns	2.55ns
2	Peak-Time	5.30ns	5.40ns
3	Rise-Time	2.50ns	2.55ns
3	Peak-Time	5.35ns	5.40ns
4	Rise-Time	2.50ns	2.40ns
4	Peak-Time	5.40ns	5.35ns

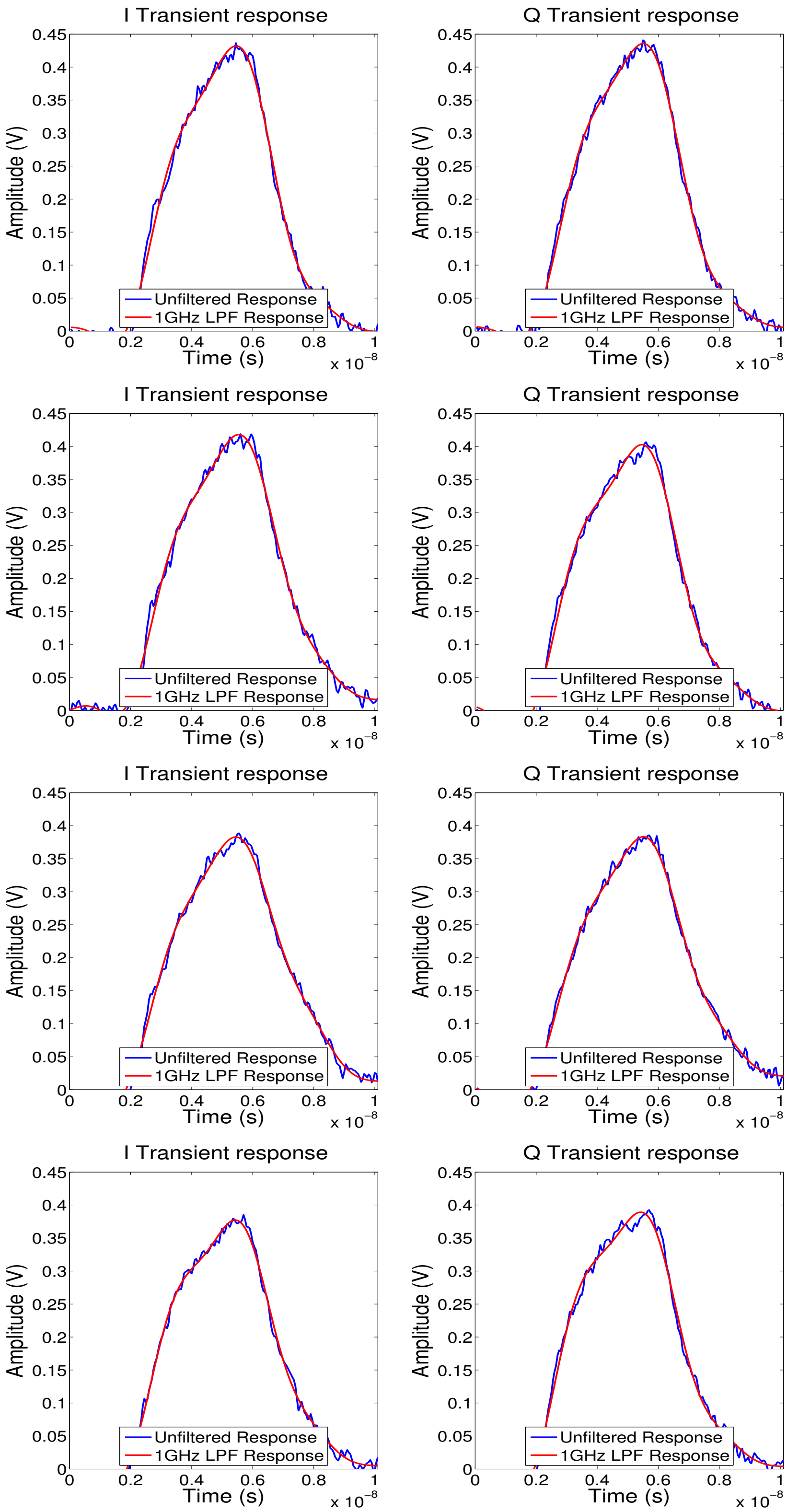

Figure 3.2: DAC 1 (top) to 4 (bottom) transient response of each I and Q output

3.1, that all of the DAC outputs report around 5.4ns. This indicates some bandwidth limitation at the DAC output. Furthermore, such a value can be used to estimate the frequency response of the system. By taking the reciprocal of the average Peak-Time measurement, the frequency response is estimated to be 180MHz. This will be confirmed in Section 3.2 relating to frequency response measurements.

3.2 Frequency Response Measurements

To generate a signal suitable for the evaluation of the frequency response, a pulse with a width equal to the clock period of the DAC source clock is again utilised. Furthermore, the DAC output is monitored using a spectrum analyser to confirm the validity of the Matlab processed results. The difference, using an absolute average error calculation, between the spectrum analyser and the Matlab processed signal is found to be $< 0.5\%$ for any single DAC output compared.

The results provided are from Matlab as, relative to the spectrum analyser, the Matlab data has a much higher resolution as a result of a much higher number of accessible data points for a given frequency range. The frequency response is obtained from a point by point division of the DAC output signal spectrum by the theoretical spectrum of the input pulse. This is a standard method to derive the transfer function of a system and is further detailed in [76]. Figure 3.3 shows graphically the resulting frequency response for the I and Q outputs of all DACs.

To ascertain if the output bandwidth is adequate, the signal attenuation at the maximum required frequency must be less than 3dB; this is the common filter attenuation cut-off value. Therefore, Table 3.2 provides the maximum dB attenuation up to the DAC clock frequency for the I and Q outputs of each DAC.

For each of the DAC outputs, the attenuation is less than 3dB which indicates satisfactory bandwidth. Any filtering that exists at the DAC output has a cut-off frequency higher than that of the maximum required frequency. It is expected that there is

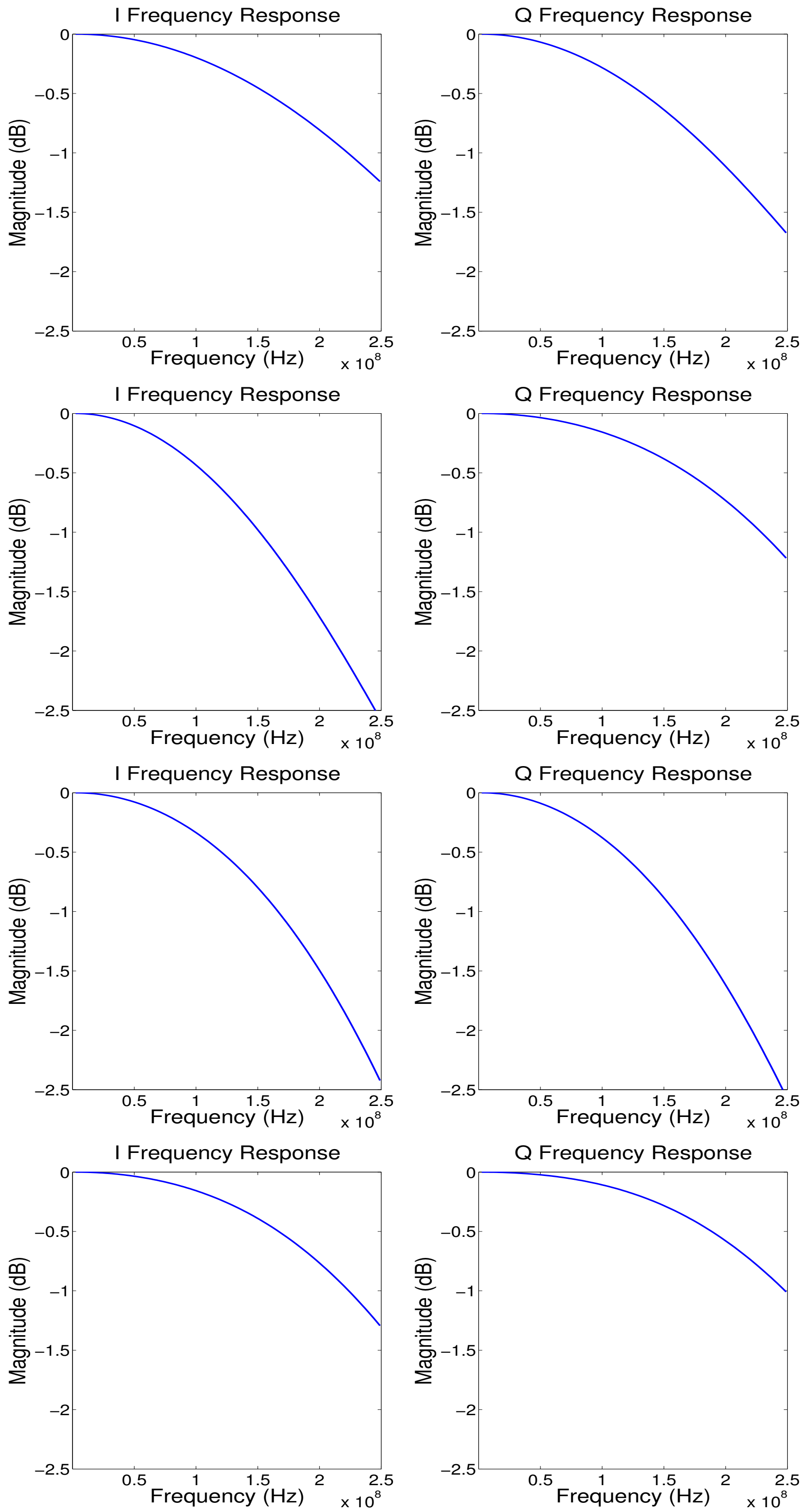

Figure 3.3: DAC 1 (top) to 4 (bottom) frequency response of each I and Q output

Table 3.2: DAC Frequency Response Results

DAC No.	DAC I	DAC Q
1	-1.24 dB	-1.67 dB
2	-2.56 dB	-1.21 dB
3	-2.42 dB	-2.55 dB
4	-1.29 dB	-1.00 dB

some unwanted bandwidth limitation, due to the differential amplifiers situated at the DAC output that artificially limit bandwidth. However, these are required to provide load balancing between the DAC differential outputs as detailed in Chapter 2.

The measurements indicate a potential problem with the Real output of DAC 2 and 3 as well as the Imaginary output of DAC 3. The measurements are different to other outputs and repeated measurement did not produce significantly different results. Further investigation is required if the output bandwidth is to be extended passed the current DAC clock frequency. Nevertheless, the dB attenuation values attained are sufficient for the current DAC clock frequency used.

3.3 Noise and Distortion

SNR and SNR Including Noise And Distortion (SINAD) are determined by applying a sine wave as input stimulus to the DAC from the FPGA. The resulting analogue signal is captured and analysed to determine the SNR, which is defined as the ratio of the fundamental signal power to that of the noise. On the other hand, SINAD is defined as the ratio of the fundamental signal power to the sum of powers of noise and distortion present, but excluding any DC element.

The digitised sinusoid used for such measurements is required to have characteristics as close as possible to an analogue signal to ensure measurement integrity. To compute such a signal within the FPGA is time consuming and resource costly; methods that enable a less costly implementation are detailed in work such as [77] [78] and [79]. As an alternative, Matlab is used to generate an appropriate sinusoid and a memory located within the FPGA, has the resulting discrete values pre-loaded at design time. Using this method, no processing is required within the FPGA. The more samples used to construct the sinusoid, the closer the representation to an analogue signal. However, this is at the expense of the amount of memory required for storage and artificially reduces the effective frequency of the sinusoid when applied to the DAC (F_{OUT}). For a fixed DAC clock frequency, the number of points which represent a sinusoid directly relates to the frequency of the sinusoid (more clock cycles would be needed to complete a single sinusoidal cycle). The system sampling clock is fixed at 250MHz (F_{DAC}). Therefore, if a single period of a sinusoid is decomposed into 256 points (N_{POINTS}) a signal at approximately 1MHz will result. The relationship between N_{POINTS} and F_{OUT} is defined in Equation 3.1:

$$F_{Out} = \frac{F_{DAC}}{N_{POINTS}} \tag{3.1}$$

Although it may be preferable to increase N_{POINTS} further (similar to an increase in sampling rate), the resulting output frequency would be too low to provide a valid measurement. This is because the clock used to load the data out of the table is also used as the source clock for the DAC. Hitherto, the frequency of the sinusoid is dictated by the number of samples it is decomposed into; frequency can be traded for accuracy of the sinusoid and *vice versa*. It is noteworthy that speeding up the DAC clock would mitigate this limitation. However, the system hardware does not support an independent DAC clock and cannot be made available without hardware modifications.

The following show the resulting SNR (Table 3.3) and SINAD (Table 3.4) for the system. A signal generator sourced sinusoid with the same characteristics as the system generated sinusoid provides a reference.

The results indicate performance very close to the maximum possible, which is determined by the DAC amplitude quantisation steps. As the DAC has a finite number of voltage steps to represent analogue signals (limited by the bit-resolution), noise

Table 3.3: SNR Results

DAC No.	DAC I	DAC Q	Reference
1	94.42 dB	94.08 dB	102.66 dB
2	92.58 dB	91.89 dB	102.66 dB
3	93.18 dB	92.74 dB	102.66 dB
4	92.22 dB	92.42 dB	102.66 dB

Table 3.4: SINAD Results

DAC No.	DAC I	DAC Q	Reference
1	95.29 dB	95.33 dB	102.65 dB
2	95.26 dB	95.25 dB	102.65 dB
3	95.24 dB	95.31 dB	102.65 dB
4	95.21 dB	95.29 dB	102.65 dB

is introduced which bounds the maximum performance. Such effect can be approximated to $6dB$ per bit, for a 16-bit device the maximum is therefore $96dB$; a value very close to those achieved practically. Furthermore, SINAD can be used to derive the Effective Number Of Bits (ENOB) [80], which is an approximation of the number of bits required to achieve a given SINAD value and is calculated using Equation 3.2

$$ENOB = \left\lceil \frac{(SINAD - 1.76)}{6} \right\rceil \tag{3.2}$$

where $\lceil . \rceil$ is the ceiling operator. From Table 3.5 it can be seen that ENOB values for each DAC are approximately 16-bit (as the value is over 15 the next integer value is 16).

Table 3.5: ENOB Results

DAC No.	DAC I	DAC Q
1	15.587 dB	15.542 dB
2	15.531 dB	15.530 dB
3	15.527 dB	15.539 dB
4	15.523 dB	15.536 dB

The signal generator reference ENOB is found to be around 17-bits, which qualifies the measurement process as capable of identifying signals with greater SINAD than that of the system. Therefore, the results attained are not artificially restricted and indicate the limit of the systems noise performance.

3.4 Amplitude Linearity

DAC linearity is characterised by Integrative Non-Linearity (INL) and Differential Non-Linearity (DNL) measurements. The former represents the difference, over the entire output amplitude, of the actual signal values against a straight line between the start and end amplitude. The latter is the full scale normalised variation in analogue value in response to a one Least Significant Bit (LSB) change in input value. In order to evaluate these parameters, knowledge of the instantaneous digital value applied to the DAC with the corresponding analogue value is required. Such a measurement cannot be performed, due to lack of digital output pin availability to attain the digital input value for a given analogue output.

However, INL can be approximated by applying, from the FPGA, an internally generated ramp signal to the DAC input. This ramp signal is derived from a counter which is incremented each DAC clock pulse, from the lowest to the highest values possible ($0 - 65535$ at 16-bit). Such values corresponding to -v_{DAC} to $+v_{DAC}$ at the DAC output. The counter provides a linear time-scale over which the amplitude can be measured.

To measure the linearity of the output ramp signal, a comparison against a straight line from the start to end amplitude over the same linear time period, is performed. The difference (defined as the error) is taken at count values over the entire range. The measurement for all DACs is shown in Figure3.4. An overall INL measurement is found by taking the largest deviation from the perfect response and is measured in Volts. Normally, INL is quantified using LSB as the unit of measure. However, this is not possible as the DACs output without the post amplification electronics cannot be isolated. The results for each DAC INL in Volts is summerised in Table 3.6. Due to the differing measurement methodology between the vendor specification and the method employed here, a specification comparison is not possible. Instead, the INL values presented here represent the measure of the uncertainty in the output voltage generated by the DACs. This data can be used to mitigate any adverse effects in the

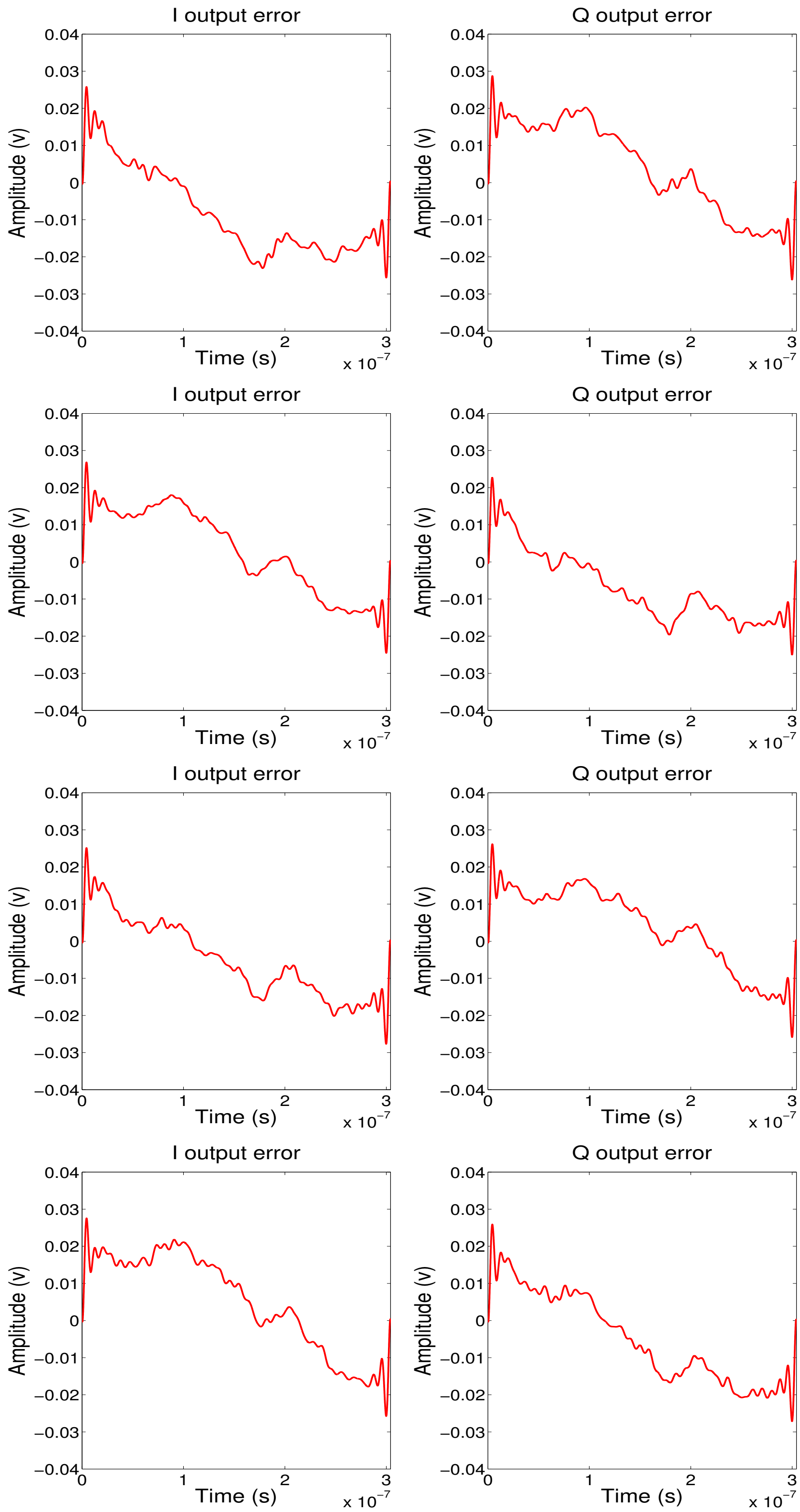

Figure 3.4: DAC 1 (top) to 4 (bottom) linearity response of each I and Q output

Table 3.6: Error for each DAC

DAC (port)	INL (V)
1 (I)	0.0258
1 (Q)	0.0287
2 (I)	0.0268
2 (Q)	0.0227
3 (I)	0.0251
3 (Q)	0.0261
4 (I)	0.0276
4 (Q)	0.0259

resulting analogue signal by applying the inverse as a calibration factor in software post processing.

3.5 Conclusions

This chapter demonstrates the successful verification of the FPGA-based platform introduced in Chapter 2. Characterisation of the system is provided for transient, frequency response, noise and distortion, and linearity by using the FPGA to generate stimuli. Standard test equipment is used to capture the resulting signal and Matlab is used to process the result. The results of all tests indicate an acceptable level of performance and the results qualify the design and development of the system and the validity of signals that the system is used to produce.

The transient response indicates an excess in output bandwidth capacity, which is confirmed by frequency response measurements. Furthermore, measurements of noise are determined as very close to the maximum possible; which is bounded by the DAC output quantisation steps. Measurements of output linearity confirms a tolerable drift over the output voltage and provides data to mitigate such effects in software if required. The characterisation performed provides an acceptable level of confidence that standard and non-standard signals generated using the system are correct. Therefore, effort can be concentrated into the implementation methods for such signals.

Chapter 4

A Simple Ethernet Stack Implementation in VHDL

Publications relevant to this chapter are detailed as follows:

Perrett, M. and Darwazeh, I. *A Simple Ethernet Stack Implementation in VHDL to Enable FPGA logic reconfigurability*, IEEE International Conference on Reconfigurable computing (ReConfig'2011), Cancun, Mexico

Ethernet is synonymous with networking and its application is ubiquitous worldwide. Its popularity is contributed to by its high bandwidth over long cable lengths and a driver-less architecture within an operating system environment. Communication over Ethernet has associated complexity, both in terms of the physical hardware necessary and the networking stack interface requirements. Hence, it is of great interest when considering a hardware implementation, of how to incorporate the required hardware blocks and network stack operations into logic hardware. The goal is the hardware implementation of a network stack, which enables the use of Ethernet to send data and control packets to reconfigure interconnected logic. By incorporating specific Ethernet hardware available within current programmable logic devices and with verification undertaken in bespoke hardware, this work represents a first attempt at such an implementation.

It is now commonplace for Field Programmable Gate Array (FPGA) vendors to offer integrated hardware blocks as part of a reconfigurable logic device. For instance, partial encapsulation of functions such as the Ethernet Media Access Controller (MAC) is available by Xilinx; in what is termed an *Embedded Core*. This differs from a *Soft Core* in that the latter's logic can be reused for other operations, whereas the former is dormant or is used only for its intended purpose.

Conversely, functions which are traditionally implemented in microprocessors, such as higher level network protocol operations known collectively as a *stack*, are not available from vendors directly. Such stacks are available from several third parties [81] [82] [83]. However, these stacks are expensive, resource costly and lack performance due to the large number of common protocols required for a generic stack.

If the aim is to receive data and control instructions over Ethernet, the required stack need not be very complex and only act upon a small fraction of protocols present in a typical network connection. This significantly reduces the design effort required and saves both cost and logic resources at the expense of design time and effort.

Such a methodology is exemplified by the work of Tian-Hua et al [84], which details a hybrid Very high speed integrated circuit Hardware Description Language (VHDL) and Matlab based method of Address Resolution Protocol (ARP) management. The work is subsequently verified via Modelsim and is targeted at improving data throughput of Gigabit Ethernet stack performance. Work by Herrmann et al [85] and earlier work by Lofgren et al [86] contain a complete networking stack for Transport Control Protocol (TCP) and User Datagram Protocol (UDP), provided in three flavours: simple (VHDL-based, UDP, No ARP), medium (VHDL-based, UDP, ARP) and complex (Mixed VHDL-Embedded processor based). Furthermore, [85] concentrates on a hybrid, VHDL-only design, similar to the aforementioned complex method.

Others, such as [87], introduce the concept of TCP-Offloading [88] [89]. Such implementations are commonly defined as a TCP offload Engine (ToE). Processes usually carried out by a Central Processing Unit (CPU) are completed in FPGA hardware, resulting in a data throughput increase and power utilisation reduction. An alternative use of a ToE, in a reduced capacity, is where only simple point-to-point protocols such as UDP are offloaded to facilitate a general purpose control interface for industrial automation [90].

The work of this chapter concerns the design and implementation of an FPGA based network stack, which considers only the minimum protocol requirements for communication over Ethernet; that of ARP protocol management. The use differs from the aims of TCP-offloading and other published work by targeting to reconfigure the operation of internal logic situated inside the FPGA, which is a requirement in many of today's communication systems. This methodology is subsequently proven in [73] for a specific application in wireless baseband signal generation. Verification of the function of the stack is provided using bespoke FPGA hardware in [91] which, to the author's best knowledge, is the principle work in this area. The design is also verified with industry standard test equipment and network analysis software.

Section 4.2 details the basic requirements for communication over Ethernet; namely the correct processing of an ARP packet. Section 4.3 details a VHDL equivalent design of a stack which is capable of receiving and replying to ARP packets, as well as enabling a data pathway for reconfigurability operations. The design is functionally evaluated in Section 4.4.1, whereby correct processing of the ARP protocol is demonstrated. The stack is then subject to a performance evaluation in Section 4.4.2, where the output of industry standard tools generates actual logic resource utilisation and maximum clock speed estimates. A critical analysis and suggestions for improvements are proposed in Section 4.4.3 and the work is concluded in Section 4.5.

4.1 Communication Over Ethernet

Ethernet fits within layers 1 and 2 of the Open Systems Interconnect (OSI) networking model summarised in Figure 4.1. Data sent over Ethernet is wrapped in an Ethernet Frame, consisting of target and sender MAC addresses and a control identifier. Furthermore, a Cyclic Redundancy Check (CRC) of the entire payload is appended to the end of each frame.

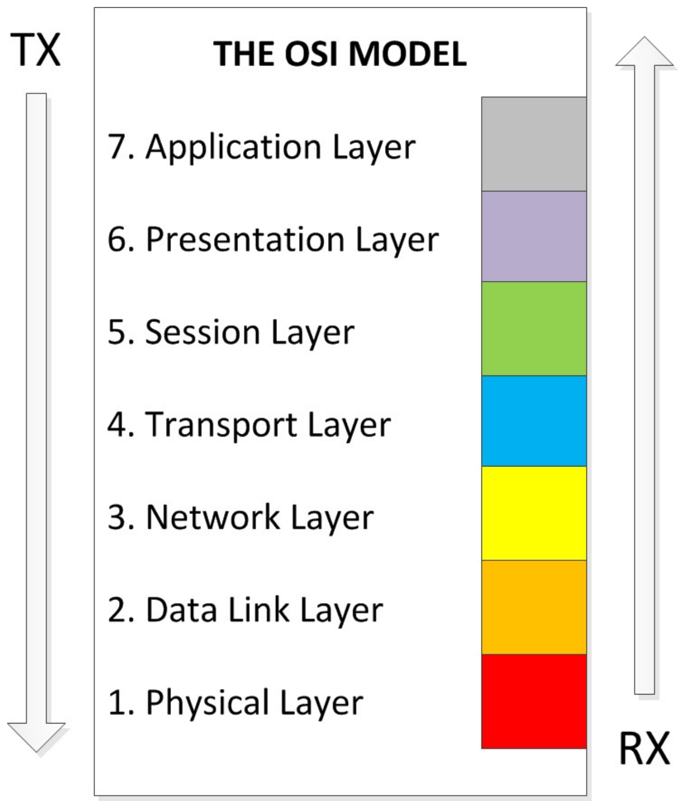

Figure 4.1: OSI networking model summary, noting direction data flows depending on receive or transmit operations

The Ethernet frame principally encapsulates other protocols and an example Ethernet frame containing Internet Protocol (IP) and either TCP or UDP at a byte-level is illustrated in Figure 4.2.

TCP is used where packet exchange over a lossy network requires guaranteed message integrity and delivery. TCP requires all packets which are received to be subjected

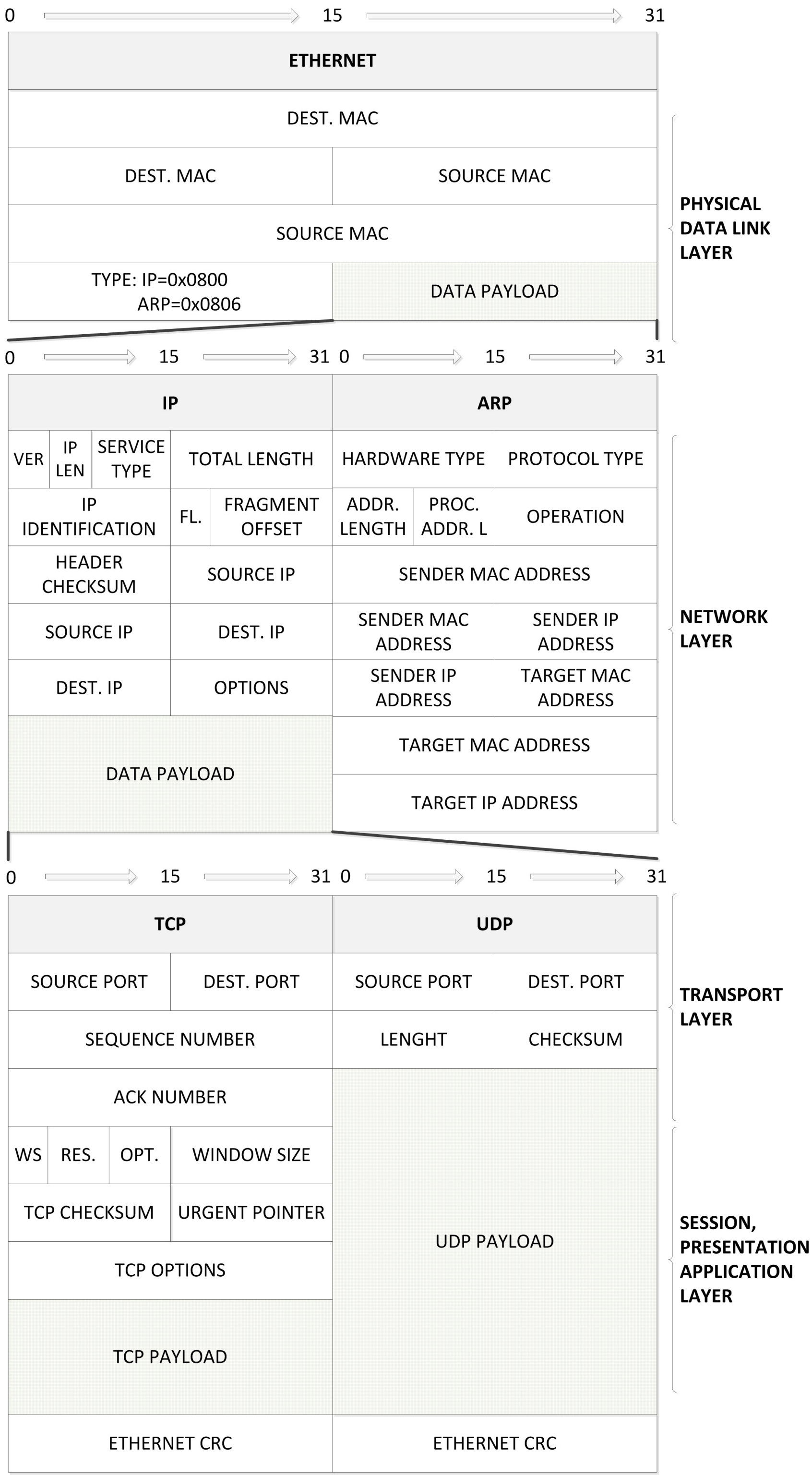

Figure 4.2: Ethernet, IP/ARP and TCP/UDP protocol byte-level configuration

to a CRC, followed by acknowledgement of correct packet reception from the target (an ACK). This is returned to the sender within the time previously specified. The ability to re-transmit a packet in case of loss requires local buffering and storage of packets in memory connected to the network device. A system which employs TCP will therefore have a large memory requirement. This effect is compounded by TCP allowing for multi-session (whereby each session will have its own retransmit buffer) as well as the duplication of control and CRC processes for each session.

Where packet loss is acceptable, for example live video streaming, an alternative protocol to TCP can be used such as UDP. UDP requires only source and destination port addresses, payload length and an optional CRC and has no requirement for packet acknowledgement. Therefore, UDP is an attractive protocol if Ethernet and the underlying encapsulated protocols are required simply to control and send data to a single connected device. Regardless of the protocol(s) encapsulated under Ethernet, adherence to the ARP protocol is required.

4.2 Address Resolution Protocol

The ARP protocol is necessary due to Ethernets reliance on statically defined MAC addresses for network routing, while network traffic is routed using network IP addresses. ARP provides a Host with a method of discovery of an Ethernet connected Target MAC address. The purpose is to link the Targets MAC address to an IP address.

A hardware implementation of ARP-only requirements is desirable for simplicity and to enable the use of the Ethernet interface to send and receive data. The reliance on a CPU-based stack and the CPU itself is removed at the expense of compatibility with other protocols. An ARP pairing process is not complicated and the transactions involved are shown diagrammatically on the left of Figure 4.3. Furthermore, a byte-level breakdown of the ARP request and reply is detailed (right part of Figure 4.3) in a format suitable for hardware implementation.

Initially, the Host broadcasts an ARP request with its own assigned IP address as the

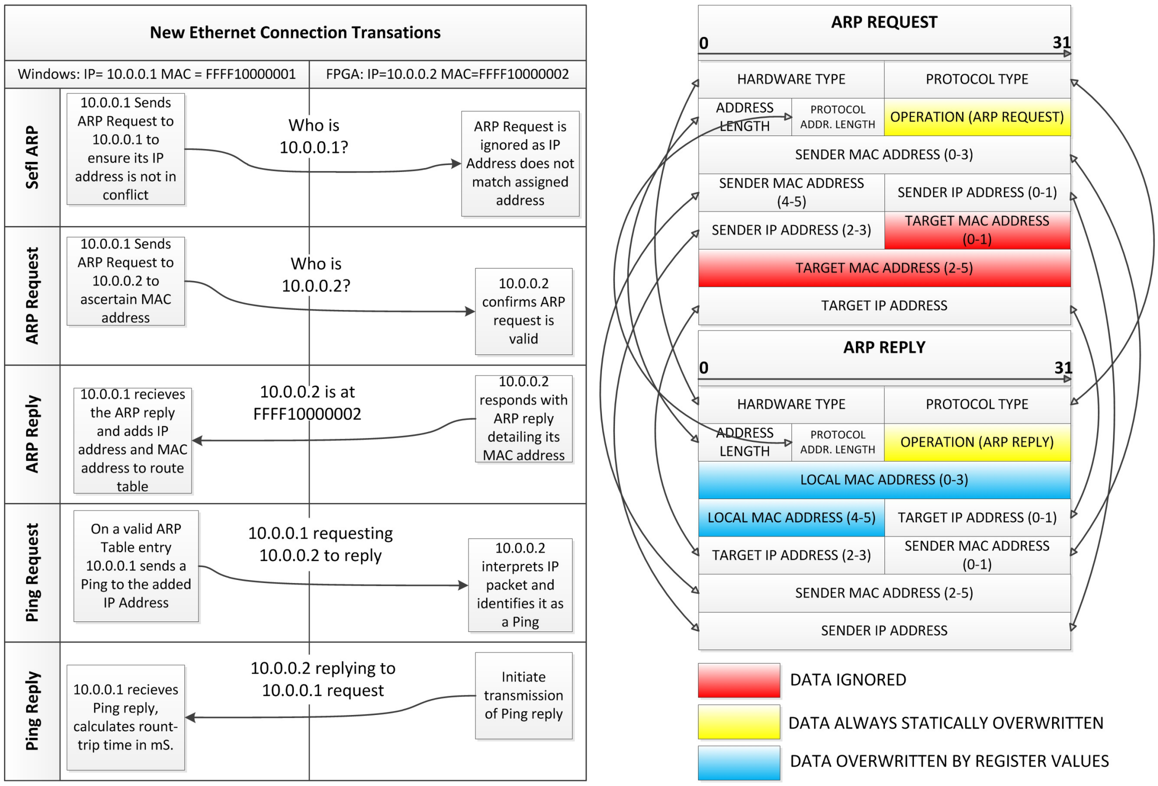

Figure 4.3: Details of ARP packet transactions (left) and byte-level description of ARP request and reply (right)

target IP address. The purpose of this so called *self* ARP is to ensure that no other device attached to the network has the same IP address as the Host. When other IP packet data from the Host commences, by the action of an IP transaction instigation by the Host, an ARP request is sent to the Target IP address.

The formatting of a request to a reply involves swapping of the Target and Host IP address, swapping of the Host and Target MAC address and overwriting the original target MAC address with the internal MAC address. As this process is the same for each ARP request, a hard-wired bus can perform the data swapping operations in parallel; reducing the request-to-reply latency.

An ARP reply is transmitted back to the Host containing its MAC address, which is linked to the original Host-transmitted IP address. This MAC/IP address pair is stored locally on the Host operating system. The ARP process is periodically refreshed and the stack must be capable of producing an ARP reply at any time.

4.2.1 Summary

Reported here is the basic requirements for Ethernet communication. A candidate for simple Ethernet packet transfer, UDP, is identified. A procedure which must be completed before any network traffic can commence, ARP, is detailed at Byte level which is suitable for hardware implementation. Hitherto, in Section 4.3, implementation of ARP targeted for FPGA hardware is detailed. Furthermore, details are provided of the use of the UDP payload for logic reconfiguration purposes within the FPGA.

4.3 FPGA Implementation

As detailed previously, the aim of this chapter is to design and implement a network stack with functionality limited to control and logic reconfiguration over Ethernet. However, there are other design considerations which form guidelines that define best practice. There are summarised as follows:

- The target Ethernet variant is 10/100 (100 base-TX). Nevertheless, as Gigabit Ethernet (1000 base-TX) is now standard in modern computers the final design should be capable of achieving the timing requirements for both speeds: 25MHz (40ns clock period) for 100-baseTX mode and 125MHz (8ns clock period) for 1000 base-TX.

- Resource utilisation should be mimimised, including the use of smallest types of logic blocks (counters instead of adders for example), by attempting to fully utilise Flip-Flop - Look Up Table (FF-LUT) pairs and by bounding variables only to the required bit-width.

- Maintain compatability with standard blocks. Vendor tools have difficulty synthesising logic that cannot be easily mapped to logic.

4.3.1 Stack Architecture

Detail of the logic and associated interconnections required to implement ARP is provided in Figure 4.4. The level of detail is not indicative of the architectural logic that is produced as a result of analysis by the vendor design tools, but instead serves as a logic representation of the source VHDL code.

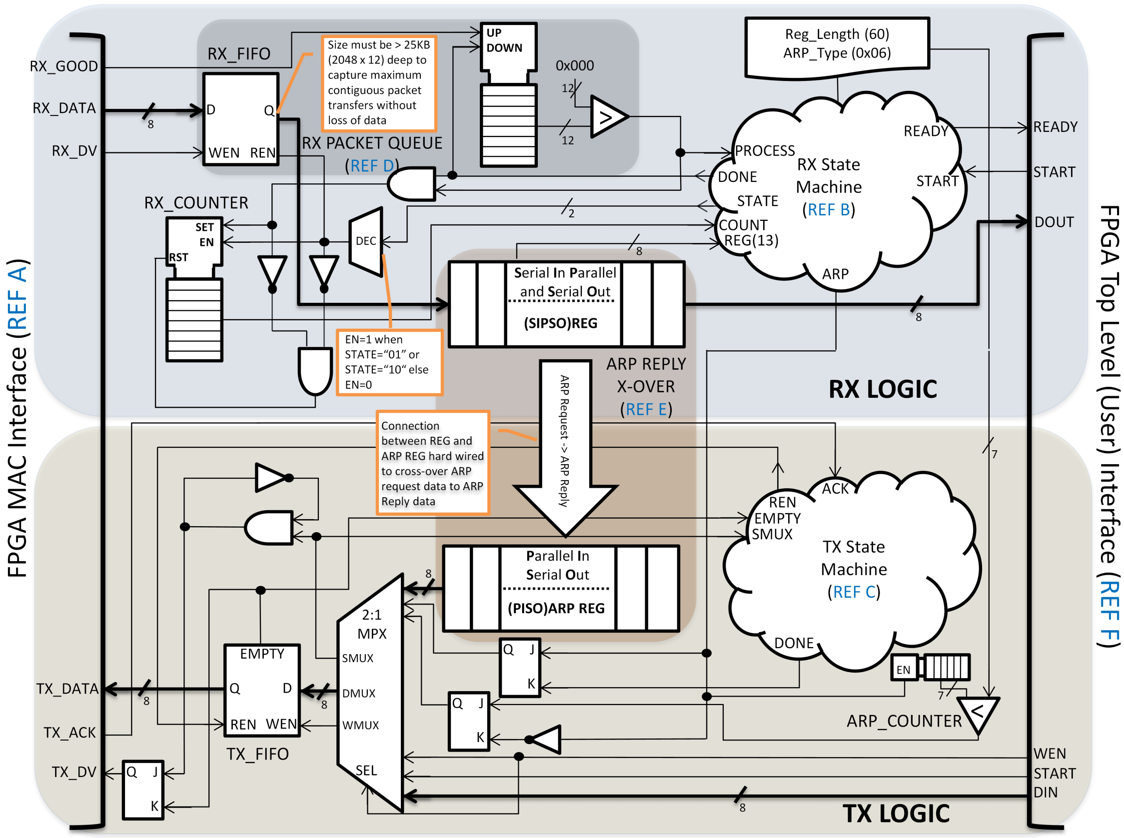

Figure 4.4: Logic Block Diagram of the Ethernet Stack operation between the MAC and the user top level

A packet, transmitted over the Ethernet hardware, arrives at the MAC (REF A), which loads the data immediately into the receive buffer (RX_FIFO). Once the entire packet has been received, the MAC signals that the packet has passed all Ethernet integrity checks by driving high the RX_GOOD signal, which is subsequently used to increment the packet queue counter (RX_PACKET_COUNTER - REF D).

A comparator at the output of the aforementioned packet queue indicates that there is a packet awaiting processing. Simultaneously, the RX_STATE_MACHINE (REF B) commences the output of 60 bytes of data. An ARP register identification byte determines if the packet is an ARP or not. If the packet is identified as ARP, it is translated to an ARP reply which requires a single clock pulse; due to the hard wired connection between the two ARP buffers (REF E). Concurrently, the TX_STATE_MACHINE (REF C) is signalled by the RX_STATE_MACHINE that an ARP packet awaiting transmission.

If the packet is not an ARP type, the IP length is read from the ARP register, before the read process is restarted by the READY signal being driven high. This informs the user a packet is valid and simultaneously presenting the first data byte at the DOUT pins (REF F). The continuation of data to the user is conditional on the START flag from the user being high, signifying that the user is ready to accept the data.

There are two pathways which can supply the transmission side of the MAC (REF A). Data and control is either sourced from the user via the WEN, START and DIN pins (REF F), or by the internal ARP reply logic driven by the TX_STATE_MACHINE. The pathways are routed using a multiplexer acting to a common MAC interface. Once the complete data payload is loaded into the transmit buffer (TX_FIFO), TX_STATE_MACHINE performs the MAC transmit procedure as defined by the Xilinx MAC specification [49]. The procedure requires the first data Byte to be presented on the TX_DATA pins and the TX_DV pin to be driven simultaneously. Further write operations are withheld until TX_ACK is driven by the MAC. The details of each **REF** are further elaborated in the following sections:

REF A - Media Access Controller interface

This refers to the standard interface as described Appendix D and 2.4.1. This is included to indicate where the standard Ethernet blocks interface to the Stack.

REF B - RX State Machine

In order to maintain synchronisation and perform packet management a State Machine is implemented. The process diagram of the state machine can be seen in Figure 4.5. In detail, the state machine loops in state A until RX_COUNTER is equal to the

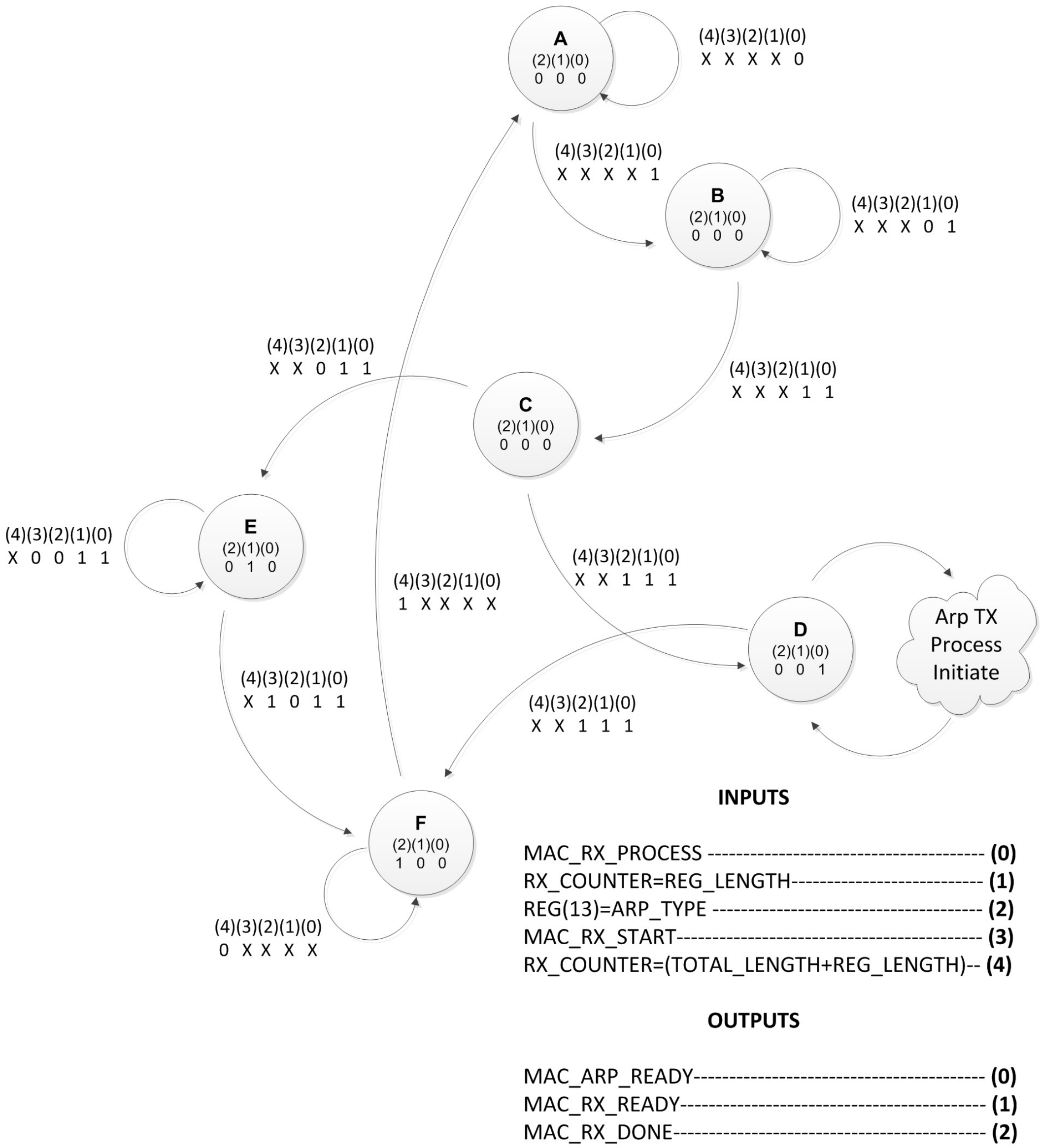

Figure 4.5: State diagram of the MAC-to-user (RX) control logic

TOTAL_LENGTH+REG_LENGTH. At this point, the state machine moves to state B but no output is driven. State B is maintained until MAC_RX_START is driven high at which point state C is entered. If the packet is identified as ARP, determined by REG(13), state D is entered and MAC_ARP_READY output is set high. If the packet is not ARP, state E is entered and the MAC_RX_READY output is set high. For the former, state D is maintained until the ARP reply transmission is complete (from the TX state machine). For the latter, state E is maintained until all data is transmitted to the user which is determined by RX_COUNTER=REG_LENGTH. Other paths lead to state F at which point the MAC_RX_DONE output is set. This state is maintained until inputs are reset at which point the state machine returns to state A, ready to process the next packet.

REF C - TX State Machine

The transmit state machine implements the standard MAC interface detailed in the vendor specification. The state diagram is shown in Figure 4.6. From the idle state

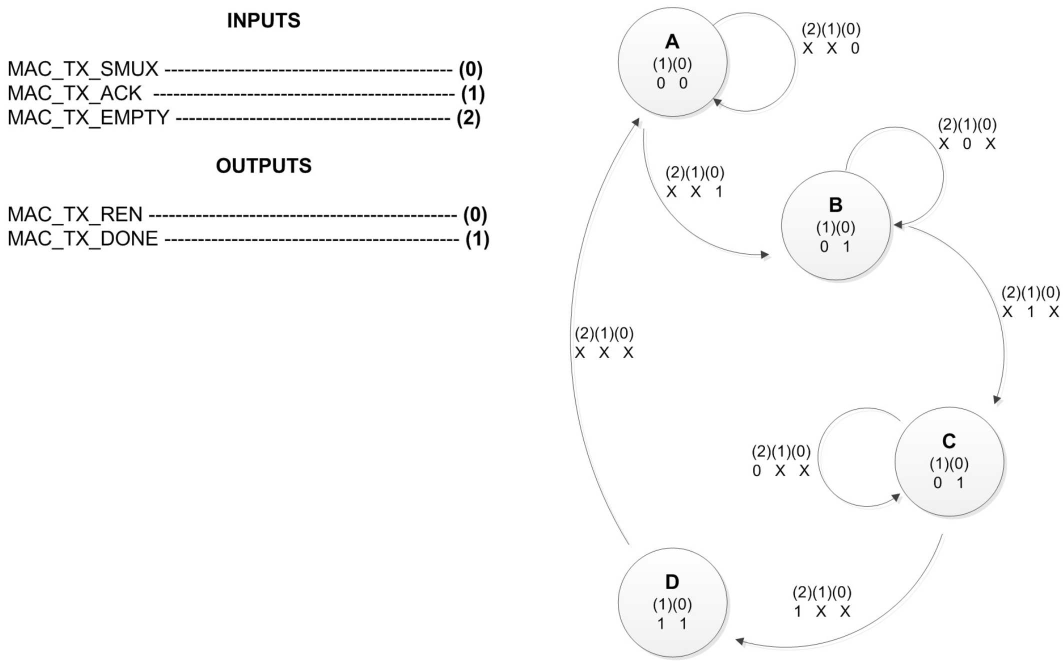

Figure 4.6: State diagram of the user-to-MAC (TX) control logic

(A), the state machine moves to state B and drives the MC_TX_REN when the MAC_TX_SMUX signal is high. No further data is written into the MAC until the MAC_TX_ACK signal is high, at which point state C is entered. State C is maintained until the MAC_TX_EMPTY signal is high, indicating that all data has been loaded into the MAC. State D is entered and the MAC_TX_DONE output is set high. This signal resets other logic before all outputs are set low and state A is returned to, ready for the next packet.

REF D - Packet Queue

An Ethernet packet of length up to the maximum data payload of 1460 Bytes can be transmitted adjacent to its neighbouring packet without any inter-packet time gap. The receiver is therefore required to process packets *back-to-back*. Alternatively, a receiver needs to buffer each packet as it arrives to be processed at a later time if, for whatever reason, the packet cannot be immediately processed and are queued. As the packet arrival order is the same as the processing order, a simple packet queue can be implemented for the counting incoming and outgoing packets.

At first, received packets are routed directly into a buffer as they arrive. Once the packet has been transmitted from the MAC and loaded into a buffer, a *good packet* flag is set from the MAC. This signal serves to instruct a counter to increment and indicates to other connected logic of a new packet in the buffer. A comparator checks the aforementioned counter value against a zero value to signal that packet(s) are present for downstream processing. The packet is read out of the buffer (using the packet length in the protocol header) and the counter subsequently decremented once processing is complete.

REF E - Address Resolution Protocol Management

This is shown for context and contains the ARP packet detection and management as described in Section 4.1.

REF F - User Logic (Top Level)

Where a packet not determined as ARP, it is instead routed to a *top level* area. The purpose of this area is detailed in Chapter 5, which details the reconfigurability of logic parameters internally to modify resulting wireless signals generated within the FPGA.

4.4 Ethernet Stack Evaluation

The evaluation of the stack is broken down into two sections, *Functional* and *Performance*. In terms of function, the Stack is verified by two methods. First, using a packet sniffer, the integrity of an ARP reply from the stack in response to an operating system ARP request, is analysed at byte level. As an additional verification stage, the operating system is interrogated to ensure the ARP reply has been interpreted correctly. Second, the function of sending a control packet for the purposes of logic reconfigurability is verified; both by analysing the data received at a byte-level using a logic analyser and then, in Chapter 5, by comparing the expected system re-configuration against the actual changes taken place.

Performance is verified by analysing the timing results of the code to ensure it will function correctly at the frequencies required. This is performed by analysis of the timing report from the vendor tools. The design is then subject to a resource utilisation analysis, both for efficient coding and in comparison to other, fully featured implementations.

4.4.1 Functional Evaluation

The objective of the functional verification is to determine the correct operation of the stack experimentally. The VHDL code that described the design is downloaded to a test platform consisting of an FPGA and the required hardware for Ethernet communication. The input to the test platform is an Ethernet connected Host computer running Windows. The Host is also running a packet sniffing application; namely *wireshark* [92].

The output of the test platform is connected to a logic analyser via dedicated debug pins which, internally in the FPGA, are connected in place of user defined logic to be re-configured. Therefore, data payloads not identified as ARP will pass through to the logic analyser for comparison against the original transmitted data payload. If a single data packet is transmitted, interception of such a packet using a packet sniffer can be synchronised with capturing of the logical output data.

The host instigates an ARP request by initiating an IP transaction; an action by which the default Host operation is to initiate an ARP request if the target MAC address is unknown. In practice, this is accomplished by the use of a *ping* command (using the windows command prompt) and by the capture of subsequent packet traffic using the aforementioned packet sniffer. To confirm the correct operation of the ARP transaction, the Host Route Table (which forms part of the Host operating system network stack) is interrogated directly with the command *arp -a* using the command prompt.

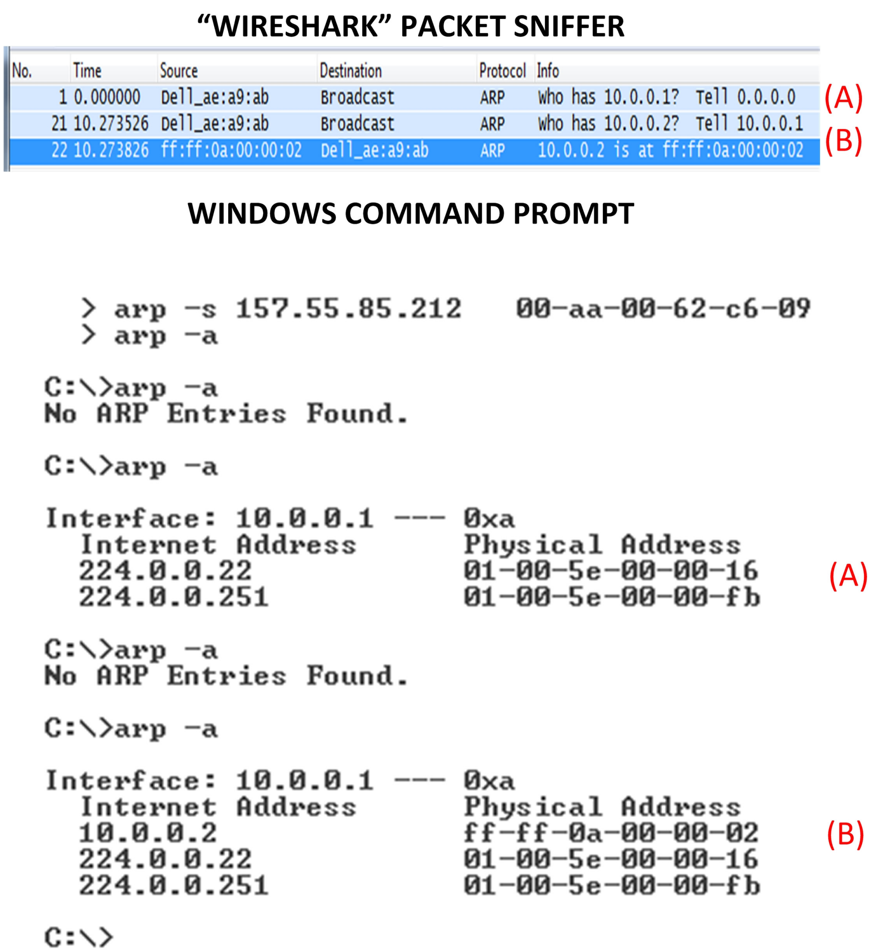

Figure 4.7: Functional verification of a successful ARP transaction between a Host and Target

Figure 4.7 displays a screen capture to illustrate the correct operation of an ARP transaction. The first line of the packet sniffer (A) shows the Host self ARP, this is supported by the first and second entry in the command prompt (also referred to as A). The former, before the self ARP, shows zero entries in the Host Routing Table. The latter, after the self ARP, shows that network traffic has been enabled as some connected devices are now present. The second line of the packet sniffer (B) shows the ARP request from the Host being transmitted to the Target, in this case at IP address 10.0.0.2. The third line shows a corresponding ARP reply from the Target to the Host; corresponding to the third entry on the command prompt that shows the successful addition of 10.0.0.2.

Recalling from Section 4.3, any packet not identified as an ARP will be directed to enact logic reconfiguration operations. As such, verification of the correct operation is completed by a second initiation of a *ping* request. This initiates transmission of a known data payload to the previously used IP address of 10.0.0.2. Figure 4.8 shows a Logic Analyser waveform capture after the aforementioned ping packet has been transmitted. The arrow demonstrates that the data sent to the Target (which is intercepted by the packet sniffer) and the data recorded and displayed by the Logic Analyser are identical. It is possible to supplement meaningful control and reconfiguration data in place of the ping data payload. Furthermore, this data can serve to change the operation of logical operations present in the FPGA hardware as demonstrated in Chapter 5.

4.4.2 Performance Evaluation

The stack latency is determined by the length of the packet (as the entire packet is buffered internally) and the system clock frequency. Additionally, a single clock cycle is required when data output is halted to determine if the packet is an ARP type or not. Therefore, unless the design is re-architected to remove packet buffering, it can be considered optimal.

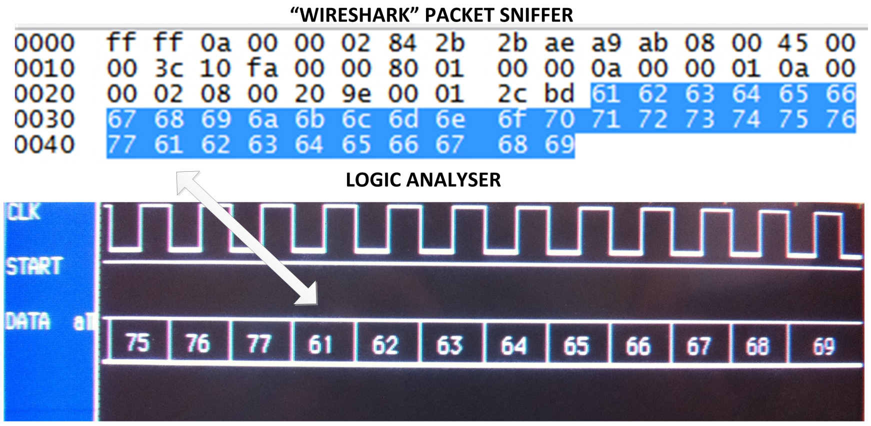

Figure 4.8: Functional verification of a packet not identified as an ARP being outputted to the user

To ascertain if the resulting design reflects the optimal resource utilisation each of the internal logic blocks is analysed. Table 4.1 shows a sample of the vendor tool logic translation. Signals are cross-referenced to those which are introduced in Figure 4.4. This indicates that the tools have correctly understood the VHDL code that describes the design. However, some of the signals have unexpected sizes which will be discussed in later sections detailing design improvements.

Table 4.1: Table detailing the resource breakdown of the stack

Resource	**Qty**	**Ref**
State Machine	2	RX/TX_STATE_MACHINE
8-bit up/down counter	1	RX_PACKET_QUEUE
32-bit adder	1	RX_COUNTER
8-bit registers	120	ARP REG
12-bit $\geq$	1	RX_COUNTER $\geq$ RX_LENGTH
9-bit $>$	1	RX_PACKET_QUEUE

Table 4.4 details the overall stack resource utilisation. It is obvious that the logic use is small when compared to the overall logic that is available in the FPGA (a Xilinx Virtex 5SX50T).

The results presented do not indicate if the design is successful at reducing the resource requirement when compared to a fully featured implementation. Therefore, it

Table 4.2: Table detailing the estimated resource utilisation from vendor tools

Resource	Utilisation	Total Available
Slice Registers	794	32640
Slice LUTs	374	32640
LUT-FF Pairs	1035	32640
Unused FF	241	1035
Unused LUT	661	1035
Fully Used	133	1035
Unique	42	1035

is prudent to perform such a comparison with a design that extends the capability to TCP. A resource utilisation comparison is undertaken to compare this work with a commercially available product from [83]. As previously mentioned, this work represents only the function required for basic communication over Ethernet and a feature comparison is provided in 4.3

Table 4.3: Table comparing the features of this work compared to a fully featured solution

Metric	This work	Fully featured solution
Re-transmit buffer	N	Y
Session Management	N	Y
ARM Management	Y	Y
UDP	Y	Y
IP	Y	Y
Partial/Fragmented packets	N	Y
Jumbo Framing	N	N
VLAN tagging	N	Y

A resource comparison of the two solutions is provided in Table 4.4, where three parameters are compared: *Slices*, *Registers* and *Look-Up Table (LUT)*. Both designs are targeted to the same FPGA device, a Xilinx Virtex 5SX50T. Registers refers to the fundamental amount of logic required to create the design; slices and LUT refer to FPGA fabric utilisation. It is shown that this work offers significant resource saving. Furthermore, Table 4.4 illustrates the overhead for an increase in TCP sessions. In conclusion, this work has a significantly smaller resource requirement compared to commercially available solutions, at the expense of functionality and protocol support.

Table 4.4: Table comparing the resource requirements of this work compared to a fully featured solution

Metric	**This work**	**4-session ToE**	**32-session ToE**
Slices	2130	4392	16433
Registers	992	5234	6012
LUTS	1923	5791	22101

The remainder of this section details the performance of this work in terms of maximum clock frequency, reported from vender tools. Table 4.5 presents two timing results. The timing constraints applied reflect the Ethernet physical-layer transfer rate required. If 10/100 mode is selected, the corresponding clock speed is 25MHz and the vendor tools can relax the clock period specification. Conversely, if the clock rate required for GbE mode (125MHz) is required, the resulting maximum clock rates increase, but with a corresponding increase in development time. The relationship between the specification set and the development time taken is based on two factors; the closeness of the required clock speed to a specified maximum (the maximum clock speed) and the number of logic elements which require synthesis (the size of the design).

Table 4.5: Table detailing the timing performance estimation from vendor tools

Signal	**10/100 @ 25MHz**	**GbE @ 125MHz**
phy tx clk	4.370nS	4.103nS
phy rx clk	4.868nS	4.868nS
mac rx clk	9.342nS	7.428nS
mac tx clk	7.96nS	6.716nS
Synthesis Time	34 Mins	72 Mins

Nonetheless, it can be seen that both timing constraints are met by the tools. It is noteworthy that targeting the design towards the larger and faster devices that are now available would lead to additional performance improvements.

4.4.3 Critical Analysis

As previously described, the design met functional, resource and timing requirements and can thus be considered sucessful. The following Sections represent a critical evaluation of the design against the guidelines stated earlier.

Latency

Currently, each packet received from the MAC must be buffered internally in full. This is due to the accumulative CRC over the entire packet by the MAC; known as *Store and Forward*. An alternative method is to initiate packet processing immediately when a packet is received on the assumption that packet integrity is good. If, after the packet is processed and outputted to target logic, a packet error is identified the receiving logic can be retrospectively signalled to discard the packet. This method is common when a low latency interface is paramount but adds significant complexity to downstream logic.

RX_COUNTER Signal

The RX_COUNTER signal is interpreted by the tools and implemented as an adder; a construct which is more resource costly than a counter [93] . Additionally, the 32-bit-width of RX_COUNTER is needlessly large as a result of the signal being instantiated as an integer. The maximum count value required is equal to the maximum Ethernet packet length of 1560 Bytes and RX_COUNTER can be truncated to 11-bits as $2^{11} = 2048$.

Flip-Flop Pairs

Table 4.4 shows that there are unused slice Flip-Flop pairs. The architecture of the FPGA is divided into slices, with each slice consists of a number of LUTs (here the term refers to a single building block of an FPGA rather than a table containing values) together with dedicated Flip-Flop logic. If the LUT is utilised but, due to design factors, the Flip-Flop remains unused this leads to wasted resource. Good design practice [94] suggests methods to minimise wasted Flip-Flops. Therefore, a future study is required to ascertain which of the current unused Flip-Flop pairs can be more efficiently used.

State Machine

Various methods exists to improve on standard state machine design [95], in particular the use of encoding methods such as *one hot* or other highly encoded types reduces logic utilisation significantly. The state machine currently used in the design is a full parallel type, representing an unoptimised implementation. As such, there is scope for optimisation using highly encoded methods.

Summary

A reduction of adder resources, removal of wasted Flip-Flop pairs, analysis of state machines and by truncating unnecessary signal bit-width will decrease the resource utilisation of the design. It is expected that there will be a corresponding increase in the timing performance as the design size is reduced. Additionally, the proposed signalling of a bad packet retrospectively will reduce the latency and increase the overall data throughput.

4.5 Conclusions

In this chapter, the design and verification of a method to encapsulate the minimum requirements for Ethernet communication into hardware is presented. The functionality of the proposed design leads to two key advantages: Firstly, it enables simple Ethernet communications by the design of a reduced feature hardware stack to manage ARP transactions. Secondly it enables packet transfer pathways to enact reconfiguration of internal FPGA logic.

The design is approached with the aim of simplicity, high performance and the low resource utilisation. Therefore, the approach uses hardware description languages instead of top level schematic methods and is based on a detailed knowledge of the logical structure required. Furthermore, the specific requirements of the ARP protocol are decomposed and as such a near optimum hardware implementation is derived.

The performance of the two key stack functions is evaluated by interrogation using packet-sniffing software and logic hardware outputs. This results in a demonstration of a successful ARP transaction followed by the correct reception of packets destined to control and reconfigure internal (to the FPGA) logic.

The design is evaluated for suitability targeted to a specific logic device (a Xilinx SX50T FPGA). This is performed by breaking down logic utilisation and by analysis of maximum clock rates. Furthermore, suggestions for future design refinements are included. A comparison to commercially available designs yields significant resource utilisation improvements at the expense of functionality.

Chapter 5 provides further qualification by using the stack detailed here to form a bridge between a user configurable Graphical User Interface (GUI) (detailed in Appendix D) and a reconfigurable wireless signal generator. Such a platform is well suited for the stack detailed here as only control and reconfiguration parameters are passed over Ethernet.

Chapter 5

Hardware Implementation of SEFDM

Publications relevant to this chapter are detailed as follows:

Perrett, M. and Darwazeh, I. *Flexible Hardware Architecture of SEFDM Transmitters with Real-Time Non-Orthogonal Adjustment*, IEEE International Conference on Telecommunications (ICT'2011), Ayia Napa, Cyprus.

Whatmough, P., Perrett, M., Isam, S. and Darwazeh, I. *VLSI Architecture for a Reconfigurable Spectrally Efficient FDM Baseband Transmitter*, IEEE International Symposium on Circuits and Systems (ISCAS'2010), Rio de Janeiro, Brazil.

Whatmough, P., Perrett, M. Isam, S. and Darwazeh, I. *VLSI Architecture for a Reconfigurable Spectrally Efficient FDM Baseband Transmitter*, IEEE Transactions on Circuits and Systems (TCAS-2), (invited paper) May 2012.

Field Programmable Gate Arrays (FPGAs) offer a unique combination of software abstraction and hardware performance, enabled by programming languages such as Very high speed integrated circuit Hardware Description Language (VHDL) or Verilog. Inherent from such languages, is a multitude of different design possibilities for a single implementation problem and significant development effort is required to arrive at an optimum solution. Nevertheless, designs which enable *real world* evaluation at *real time* speeds of algorithms, or operations ordinarily restricted to simulation environments are possible. In this chapter, an FPGA implementation of a method of generating non-orthogonal Frequency Division Multiplexed signals is presented, where the spacing between sub-carriers can be controlled externally. The internal data-paths and associated logical operations within the FPGA, are constructed so as to react to changes which dictate the aforementioned spacing and as such represent a dynamic transmission platform for research purposes.

FPGAs are regarded as the apotheosis of flexible, yet high performance environments for building logic systems. Logic systems spanning millions of gates are possible that exceeds the computational capabilities of similar scale Central Processing Units (CPUs). Where a system is based on standard logic blocks, such as the Inverse Fast Fourier Transform (IFFT), it is possible to arrive at a highly optimised solution using FPGA vendor licensed Intellectual Property.

Orthogonal Frequency Division Multiplexing (OFDM) has proven to have robust performance in multi-path fading channels [96] [97]. A contributing factor is the nature of the resulting signal, whereby input data streams are modulated into a superposition of sub-carriers. Inherent in OFDM is a reduction in the overall data throughput available in a given channel; each OFDM symbol requires slightly higher bandwidth than equivalent single-carrier methods. Hence, there is considerable research interest in methods to reduce the required bandwidth, while still maintaining backward compatibility and wireless performance characteristics of OFDM .

Orthogonal sub-carrier spacing is the minimum possible to result in zero Inter Carrier Interference (ICI) in the resulting carrier matrix. A reduction in the separation of sub-carriers removes this quintessential property and introduces interference between the sub-carriers. It is therefore of significant benefit to garner a degree of control over the sub-carrier spacing reduction, so as to eschew unwanted interference. The noise characteristics of a given channel can also be considered as a limiting factor in the amount of interference that can be endured whilst maintaining detectability. A noisy channel can potentially fail to maintain signal integrity with a negligible compression of sub-carriers. Therefore, it is an advantage be able to dynamically control the sub-carrier spacing on a symbol-by-symbol basis, as a reaction to potentially varied channel characteristics.

If non-orthogonal signal generation is required, the implementation complexity is increased. For a transmitter, any increase is modest depending on the methodology employed; for the receiver, complexity is increased dramatically. Therefore, a key goal is to have a controllable transmission system where additional complexity can be afforded and enables the detection of the subsequent signal to be as simple as possible. This may include the dynamic modification of the transmitted signals sub-carrier spacing, to match the current channel characteristics and hence aid detection. Parallels can be drawn between this methodology and previous work on adaptive modulation for OFDM [13].

It has been shown that the implementation of an OFDM transceiver in an FPGA is perfunctory using standard Discrete Fourier Transform (DFT) blocks. Such work has been demonstrated in [8], [12], [11], [10] and has been subject to improvements in terms of multi-band operation [9] and implementation optimisation [25] and clocking [7]. Additional features which are specific to FPGAs, such as adaptive modulation based around estimated channel performance [13], can offer throughput enhancement when compared to traditional architectures. All designs share commonalities of standard logic block utilisation, in this case the IFFT for transmission and the Fast Fourier Transform (FFT) for symbol recovery. Also in common is the orthogonal sub-carrier spacing employed, which is where similarities with the work of this chapter diverge.

This chapter details the implementation, in an FPGA using VHDL, of a Spectrally Efficient Frequency Division Multiplexing (SEFDM) signal generation method based on [98] and [99]. The purpose of this work is to implement an SEFDM generation platform on an FPGA and evaluate the signal generated practically. The use of standard logic blocks maintains compatibility and familiarly with current OFDM implementations, which are attractive as potential candidates in future wireless communication standard proposals. It is of interest to have the ability to modify the aforementioned sub-carrier separation, to achieve functionality similar to current Matlab simulations. Furthermore, SEFDM signal sources are required to evaluate FPGA implementations of non-orthogonal detection techniques, which are specific to SEFDM [100] [101]. It is also desirable to evaluate other implementations of non-orthogonal signal detection methods which may, in the future, be tailored for SEFDM detection [102] [15]. The implementation is purposely targeted at a general case where, within available resource limits, the values by which the non-orthogonality is derived are unbounded. This requires the implementation of methods to derive the SEFDM signal at an algorithmic level. Although flexible, this implementation methodology tends towards high resource utilisation which is subject to detailed analysis.

Considerations of resource versus logic operational speed is discussed, which, in the author's opinion, is the principle design factor in any FPGA implementation. This discussion extends to techniques for increasing operational speed, to the detriment of system latency, through techniques such as pipelining and combinatorial logic. A functional demonstration of the implementation using both simulation tools and with real signals generated within an FPGA, controlled via a custom Graphical User Interface (GUI), is provided; this constitutes the first time such signals have been created and transmitted via an FPGA.

Section 5.1 provides a theoretical overview of non-orthogonal generation techniques; namely Faster-than-Nyquist (FTN) (Section 5), Direct Digital Synthesis (DDS) (Section 5.1.2) and SEFDM (Section 5.1.3). This is followed in Section 5.1.4, by a complexity estimate comparison of the latter two methods, to ascertain the potential reduction in complexity offered by SEFDM. Section 5.2 details the implementation of the SEFDM concepts introduced in the preceding section(s). Furthermore, an overall description of the method employed to implement SEFDM into an FPGA is detailed. Consideration is given to resource utilisation of specific portions of the design, such as modulo arithmetic (Section 5.2.1), output phase shift generation (Section 5.2.3) and the resource effects of a change in the sub-carrier compression (Section 5.2.4). A study into data value scaling, internal to the FPGA, is discussed and critical areas identified in Section 5.2.5.

The functional performance of the SEFDM implementation, using source code simulation methods is detailed in Section 5.3. This results in simulated spectral plots for different sub-carrier compression values. Additionally, an evaluation of the design resource utilisation is performed in Section 5.4. This section also provides confirmation of the theoretical complexity detailed in Section 5.1.4, using actual FPGA resource data for comparison. Furthermore, the design is tested with an FPGA device and practical results of SEFDM signal spectrum are presented in Section 5.5. A critical evaluation of the design in terms of resource and performance is provided in Section 5.6. The work is concluded and further areas of research are identified in Section 5.7.

5.1 Non-Orthogonal Modulation Techniques

In this section, alternative methods for the generation of non-orthogonal sub-carrier generation which has been successfully implemented in an FPGA are reviewed. In principle, all methods (including SEFDM) generate a signal with similar characteristics, but the methods of generation differ greatly. This section serves as an overall descriptive appraisal, rather than a complete analytical evaluation of different techniques. Only methods which have been previously implemented in an FPGA, specifically for non-orthogonal signal generation are included. Therefore, High-Compaction Multi-Carrier Modulation (HC-MCM) [23] and Overlapping Orthogonal Frequency Division Multiplexing (OvFDM) [24] are excluded.

5.1.1 Faster-Than-Nyquist Method

The FTN concept was introduced in 1975 by Mazo at Bell Labs [103]. At the time FTN signal generation was considered too complex and in the context of an apparent abundance of available bandwidth against demand, implementation was not pursued. However, recent demand for bandwidth has vastly exceeded the cost of introducing complexity to a communications system, due in part to the low cost per transistor offered from devices such as FPGAs and Application-Specific Integrated Circuits (ASICs). In particular, the system exemplified in [104] builds on the earlier Mazo work; purposefully mirroring a traditional OFDM communications system by incorporating an IFFT.

An alternative to rectangular orthogonal basis used in OFDM [12] is used and defined as Isotropic Orthogonal Transform Algorithm (IOTA). It has been shown in [105], [106] and [107] that using IOTA bases offers improved localisation in both time and frequency over rectangular, IFFT derived basis. IOTA has an additional benefit of removing the need for a guard period; used with OFDM where Bit Error Rate (BER) improvement is sought. An IOTA pulse is derived by orthogonalising a Gaussian pulse in both time and frequency. An important property of the IOTA pulse is the equivalence of time and frequency domain representations and therefore, channel propagation affects each domain symmetrically.

IOTA can only guarantee orthogonality for *real* values (so called *weak orthogonality*) and as a result traditional Quadrature Amplitude Modulation (QAM) is prohibited. Instead, Offset Quadrature Amplitude Modulation (OQAM) is utilised, (defined as OFDM/OQAM), which provides *real* only values at time-spacing of half that of QAM. To elucidate further, for every complex QAM value at time (τ), two *real* OQAM values are presented, each at $\frac{\tau}{2}$. In [108], it is shown that the modification of an IFFT based modulation system to one capable of producing IOTA pulses is trivial, by the application of time-domain poly-phase filter banks, while compatibility with OFDM is maintained. It is also shown in [107], that these filters add negligible computational overhead when compared to the complexity of the IFFT, although this effect is not quantified.

While OFDM/OQAM does not result in a bandwidth saving compared to OFDM, (unless OFDM with a guard band is considered), a reduction in required bandwidth utilisation can be realised if the pre-modulated signal is subject to modification in the time domain; the improved Inter Symbol Interference (ISI) and ICI performance offered by IOTA is exploited. This operation is defined as FTN mapping.

Adjacent time symbols are summed and the resulting products are mapped onto an appropriate IOTA function. Each FTN symbol will contain elements of neighbouring symbols, the amount of which can be user defined and is directly related to the bandwidth saving. The ensuing ISI and ICI is mitigated by the use of IOTA function and detectability is maintained below the Mazo FTN limit [109].

The current FTN implementation is at an advanced stage, whereby hardware resource improvements [108] by the use of pre-calculated co-efficient stored in Random Access Memory (RAM), together with developments on FTN specific detector techniques [110] have led to a transceiver implementation using FPGA hardware. This design is subject to a performance evaluation in wireless channel environments in [111]. The main limitation of FTN is in the choice of Orthogonal Base which represents a fundamental limit on modulation schemes which can be employed, severely limiting the use of FTN in high data-rate applications.

5.1.2 Direct Synthesis as a Bank of Modulators

A classical method of multi-carrier signal generation is by the use of a bank of analogue signal modulators. Each signals phase, amplitude and frequency can be controlled independently. The analogue modulators can be replaced by DDS, which generates sine and cosine vectors without introducing computational complexity at the expense of memory utilisation. DDS removes almost all arithmetic operations through the use of a counter based *Phase Accumulator* and a RAM based *Phase-To-Wave Converter*.

Phase Accumulator

The Phase Accumulator is a logical counter with a maximum count which represents the maximum phase, θ_{MAX}, which is determined by:

$$\theta_{MAX} = 2\pi = 2^N \tag{5.1}$$

Phase is incremented in Δ_{ACC} steps every T_S period, which directly relates to the required frequency, F_O. Hence, 2^N is reached in time T_O, where

$$T_O = \frac{1}{F_O} = \frac{2^N T_S}{\Delta_{ACC}} \tag{5.2}$$

and

$$\Delta_{ACC} = \left\lfloor F_O \frac{2^N}{F_S} + 0.5 \right\rfloor \tag{5.3}$$

where $\lfloor . \rfloor$ is the floor operator. In terms of frequency, the minimum tuning step, $\Delta_{F_{OMIN}}$ is found by

$$\begin{aligned}
\Delta_{F_{OMIN}} &= F_O(\Delta ACC + 1) - F_O(\Delta_{ACC}) && (5.4)\\
&= \frac{F_S}{2^N}(\Delta_{ACC} + 1 - \Delta_{ACC}) && (5.5)\\
&= \frac{F_S}{2^N} && (5.6)
\end{aligned}$$

and it can therefore be determined that the maximum frequency step, $\Delta_{F_{OMAX}} = F_S/2$, which, although satisfies the Nyquist criteria is often substituted with $\Delta_{F_{OMAX}} = F_S/4$. N thus determines the frequency tuning steps and is defined as:

$$N = \left\lfloor Log_2\left(\frac{F_S}{\Delta_{F_{OMIN}}}\right) F_O \times S \right\rfloor \tag{5.7}$$

Phase-To-Waveform

The conversion of the accumulated phase values to a periodic waveform is accomplished by addressing pre-calculated sine and cosine co-efficient values; stored in a RAM based Look-Up Table (LUT). It is often desirable, in situations where a chosen value of N is large, to reduce the number of LUT values with a corresponding reduction in the memory space required. A common value of N is 32, which would therefore require an exorbitantly high number (N^{32}) of memory locations. LUT cardinality is therefore reduced to 2^P, where $P \leq N$. Each value stored in the LUT can be derived by:

$$LUT(i) = \left\lfloor a.(b + sin^{-1}(\frac{2\pi i}{2^P}) + 0.5) \right\rfloor, 0 \leq i < (2^P - 1) \tag{5.8}$$

where b is the offset and a represents the peak amplitude. Unlike COordinate Rotation DIgital Computer (CORDIC) (further details are provided in Appendix F), DDS is subject to digital noise effects, phase jitter, phase quantisation and amplitude quantisation.

Phase Jitter: Discrete systems which are clocked can only operate when coinciding with a clock edge, so called synchronous operation. If the synthetic DDS output signal does not coincide with a clock edge, the resulting signal must instead slot into an adjacent clock; this results in phase noise. This effect can be circumvented by the application of diligent DDS design methodologies, such as ensuring the output frequency is a multiple of available system clocks.

Phase Quantisation: Recalling earlier, whereby N is reduced to P it can be concluded that, due to a reduction in the number of available phase values for a given phase accumulation, a non-optimal periodic waveform will result. This is often defined as *spurs*.

Amplitude Quantisation: As with all digital representations of analogue signals, amplitude must be quantised into 2^M levels, where M is the bit-width representation of the data. This effect is quantified as Signal to Noise Ratio (SNR). However, in the case of DDS the effects of Spurious Free Dynamic Range (SFDR) will always have greater influence on the output signal so long as $P = M + 1$ (P is the LUT address range introduced in Section 5.1.2). It is therefore of considerable interest to reduce the effects of phase quantisation where possible *.

*An excellent guide to techniques for reducing phase quantisation is found in [112]

5.1.3 Spectrally Efficient Frequency Division Multiplexing

SEFDM signals may be defined as a general class of multi-carrier signals, where the sub-carrier frequency separation results in a non-orthogonal set of modulated carriers. Such signals have been of interest recently as means to create communication systems offering bandwidth saving [91]. Mathematically, these signals may be generated by modifying typical OFDM signal generation methods, in structures similar to the one shown in Figure 5.1 [22].

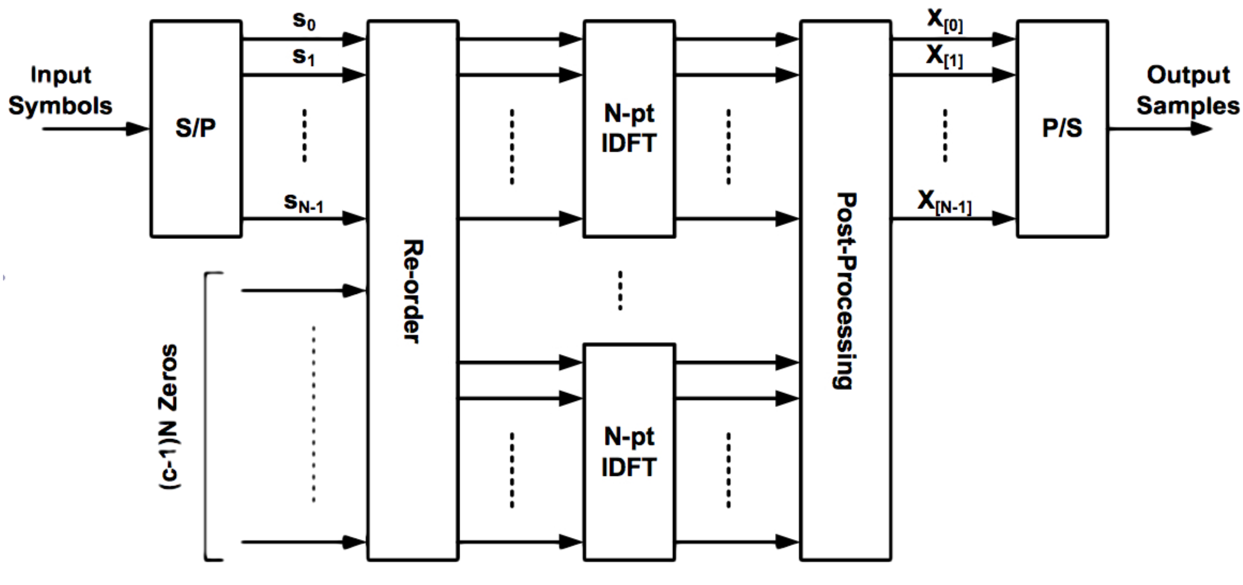

Figure 5.1: SEFDM based transmitter

The output signal samples, $X[K]$, are functions of two parameters; the number of sub-carriers, N and the relative frequency separation of the sub-carriers, $\alpha = 1$, which also defines the bandwidth saving relative to OFDM. For implementation purposes, α is taken as the ratio of two integers b and c. As such, $X[K]$ normalised may be represented as

$$x[k] = 1/\sqrt{N} \sum_{i=0}^{c-1} e^{\frac{j2\pi ib}{cN}} \sum_{l=0}^{N-1} s_{(i+lc)} \, e^{(j2\pi lk/N)} \; 0 < k < N-1 \tag{5.9}$$

where $S_0 - S_{N-1}$ are the input symbols. The resulting system will consist of c, N-length Inverse Discrete Fourier Transform (IDFT) blocks. N source data symbols are padded with zeros or repositioned, as a function of a modulo operation in base of b, to form a new matrix of size cN. The implementation in an FPGA of an IDFT is resource intensive and it is standard practice to use IFFT where possible. N is therefore restricted to values falling at 2^L, where L is an integer greater than one.

5.1.4 Computational Complexity Comparison

The aim of this section is discover multi-carrier signal generation methods that are appropriate for a specific number of sub-carriers (N) and sub-carrier spacing (α). The complexity growth, using Big $\mathcal{O}$ notation, of two method of SEFDM generation (defined as Type 1 and Type 2 [113]) is included. Type 1 is generated using a single large IFFT of size N/α and Type 2 is generated using c, N-length IFFTs. An equivalent design based on DDS as the sub-carrier source is included for comparison. FTN is not included in the comparison, as the literature does not provide a complexity growth estimate for the design.

The Big $\mathcal{O}$ estimates for DDS (N^2), Type 1 SEFDM $((\frac{N}{\alpha}).Log_2(\frac{N}{\alpha}))$ and Type 2 SEFDM $(c.(N.Log_2(N))$ are plotted in Figure 5.2. N is the number of points and α the sub-carrier spacing fraction as a product of two values; namely b/c. Additionally, OFDM is plotted as a baseline reference.

It can be seen that Type 1 SEFDM has lower complexity than Type 2 SEFDM, for a given value of α. However, the size of the IFFT required for Type 1 SEFDM is derived from N/α. This often results in a non integer or non power of two value, both of which are required for hardware implementation. It can also be seen that, when both a small value of N and a large value of c are selected, DDS may be a preferable method over any type of SEFDM.

Figure 5.3 provides the complexity impact when the value of c is raised for a Type 2 transmitter. The value of c dictates both the bandwidth compression factor (α) and the granularity of the sub-carrier spacing. It is plotted against DDS (which has

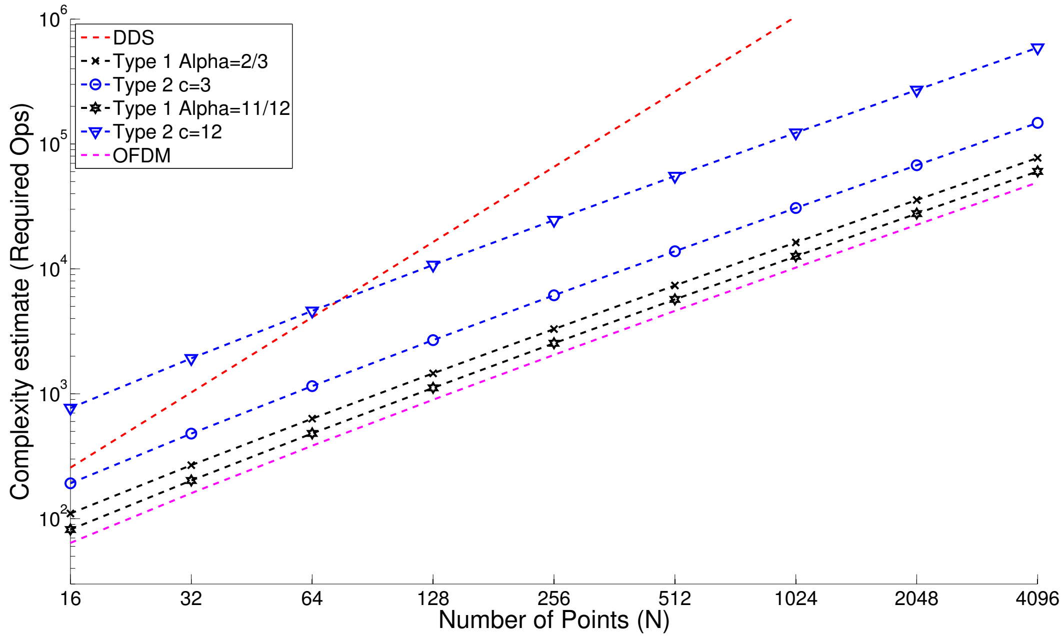

Figure 5.2: SEFDM (Type 1 and Type 2) Big $\mathcal{O}$ complexity analysis against the DDS method. OFDM based on a single IFFT is also included for comparison.

no such limitation) and OFDM as a baseline reference. It can be seen that there is

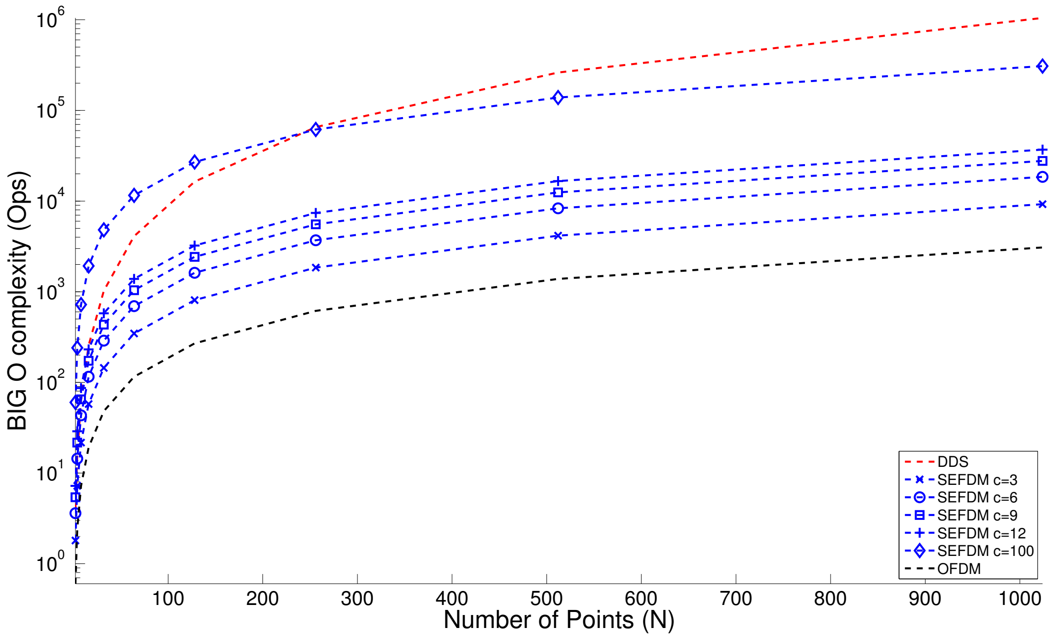

Figure 5.3: SEFDM (Type 2 only) Big $\mathcal{O}$ complexity analysis compared with DDS. Various sub-carrier granularity values are shown

an intersection between DDS and SEFDM at which point the complexity reduction offered by SEFDM is reduced to 0 ($c = 100$). A large value of c would provide a very small granularity between sub-carriers and is not a realistic scenario. Nonetheless, above 512 sub-carriers SEFDM always reports a lower complexity compared to DDS. It can be concluded that, when compared to the DDS method, both SEFDM methods (Type 1 or Type 2) offer significant complexity reduction and there exists enough scenarios where the SEFDM method is superior to justify the work in this Chapter.

5.2 SEFDM Implementation Targeted for FPGA

Recalling from Equation (5.9), the spacing of the sub-carriers is dependant on two values, b and c. Where α is defined as $\frac{b}{c}$. OFDM is defined as the case when $b = c$ and $\alpha = 1$. Any value where $b < c$ would result in $\alpha < 1$ and a subsequent reduction in the sub-carrier spacings. The stages required to create an SEFDM signal based on Equation (5.9) can be summarised as:

- The data symbols, S_n, are reordered by a modulo arithmetic function. The data symbols are placed in positions that are integer multiples of b, with zeros placed in other positions to form a zero padded vector, of size cN.

- A Column-Major-Reorder process is applied to the zero padded vector to form a matrix of size $c \times N$, with each N-length row assigned to an IFFT block. There are c number of such blocks in total.

- Post IFFT phase rotation is applied by multiplication with frequency shift vectors, derived from b and c using a IDFT.

This process is shown diagrammatically in Figure 5.4, which illustrates the operation of the procedure required to generate SEFDM.

Conceptually, this procedure is mapped to equivalent logical operations and connected to the Ethernet interface detailed in Chapter 4. Figure 5.5 provides a graphical view of the topology that exists within the FPGA to generate SEFDM. The top

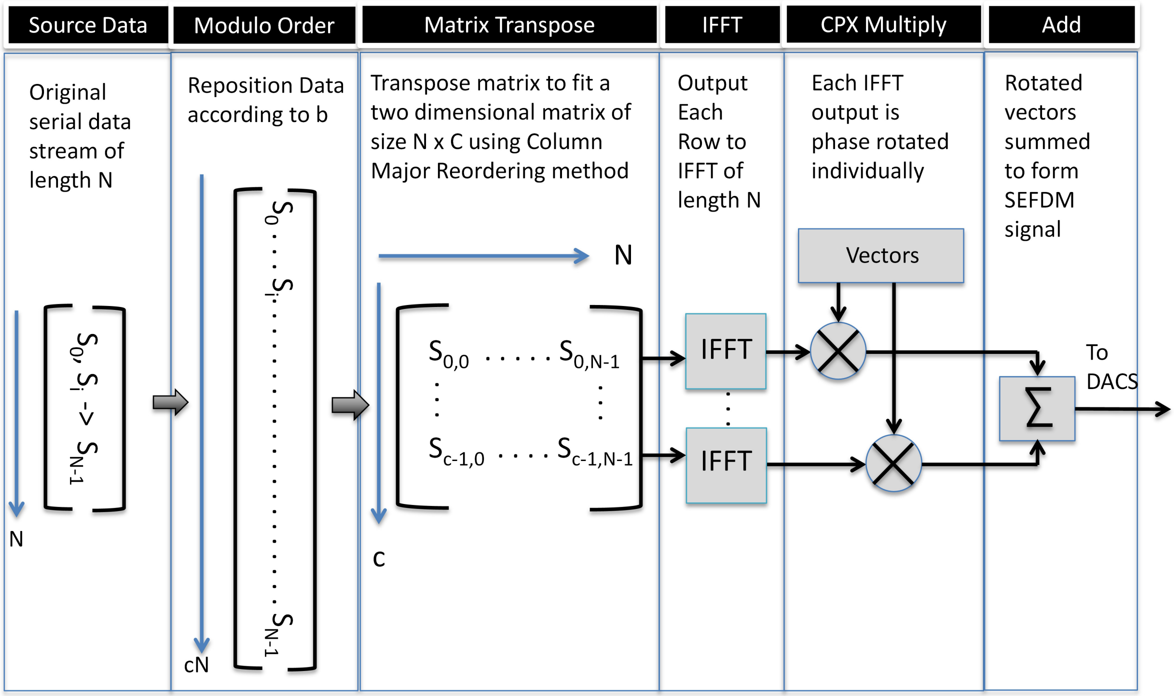

Figure 5.4: Block Diagram of SEFDM transmitter operation block diagram

left block is the aforementioned Ethernet interface to the user GUI which is detailed in Appendix D. The block below is where source data (from the user over Ethernet) is modulated and ordered into a zero padded matrix using modulo arithmetic. The block in the centre is the SEFDM IFFT block, where the output of each IFFT is modulated with vectors generated from an IDFT. The source data for the IDFT is an identity matrix, of size $cN \times c$. The final SEFDM symbol is created by summing all of the IFFT outputs together.

5.2.1 Modulo Arithmetic Implementation

The re-ordered position of the source symbols if identified using a natural ordered incrementing integer when equal to a multiple of the value of b. A naive implementation of this is achieved by the division of a integer counter with b. The quotient value of the divider is used as the index position of the source symbols. The index is used only if the division remainder is equal to zero, otherwise a zero value is applied. This process is shown logically in Figure 5.6.

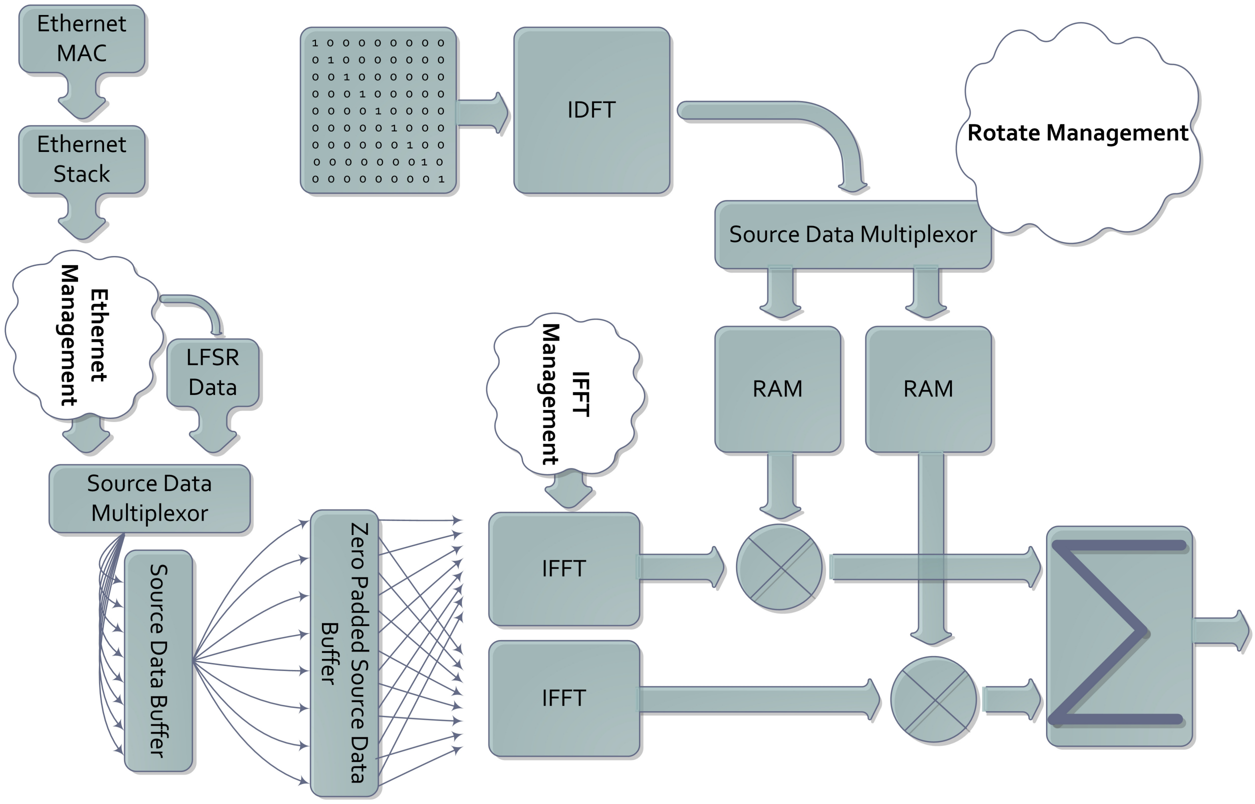

Figure 5.5: Logic topology diagram of an SEFDM transmitter

In detail, a counter with value 0 to $cN-1$ is used as the input to a divider and a one-hot encoder. The purpose of the divider is to provide indication when the count value and the value of b are the same. This is performed by a comparison between the instantaneous result of the divider, where the division value is ignored (and instead replaced with a null value) unless the remainder of the division process is zero. This indicates that the result of the division is a factor of the value of b. The output of the division indicates the next valid input data value to be loaded into the next valid d-type flip-flop, specified by the output of the one-hot encoder. The one-hot encoder also uses the count value as its input and is used to generate enable signals for the d-type flip-flops. The number of outputs of the one-hot encoder is dictated by the maximum count value. For a maximum count value of $cN-1$, the required number of bits to represent this would be $Log_2(cN-1)$. This would be converted by the encoder to cN number of independent enable signals to cN number of Independent flip-flops.

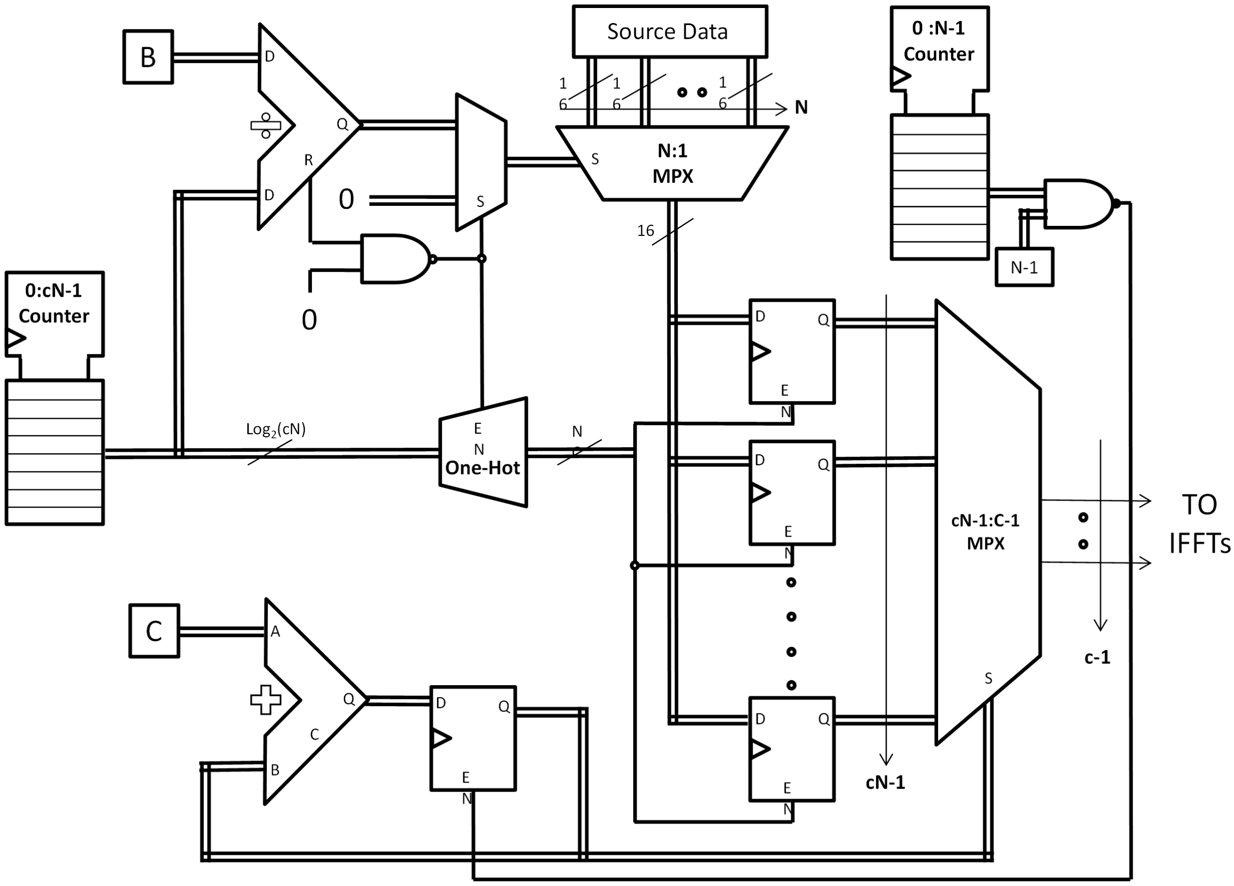

Figure 5.6: Logic diagram of Modulo divider circuit

The output of the d-type flip-flops is multiplexed from size cN to size c, N-times. This is analogous to a column major reordering process. A counter with a value range of 0 to $N-1$ is compared to the value of N, which is repeated c-times. This creates a counter which moves between 0 and N, c number of times. The output of the multiplexor is used as the input to c, IFFT blocks which require data serially (from 0 to $N-1$).

An equivalent operation is to copy source symbols, indexed by a natural counting integer, to positions indexed by a second integer incrementing by factors of b. All other positions of the destination matrix are set to zero. For example, if data symbols (S_n) are subjected to reordering based on $b = 2$, the resulting matrix (ϕ) will be conditionally copied as $S_0 \rightarrow \phi_0$ and $S_1 \rightarrow \phi_2$. This is shown logically in Figure 5.7.

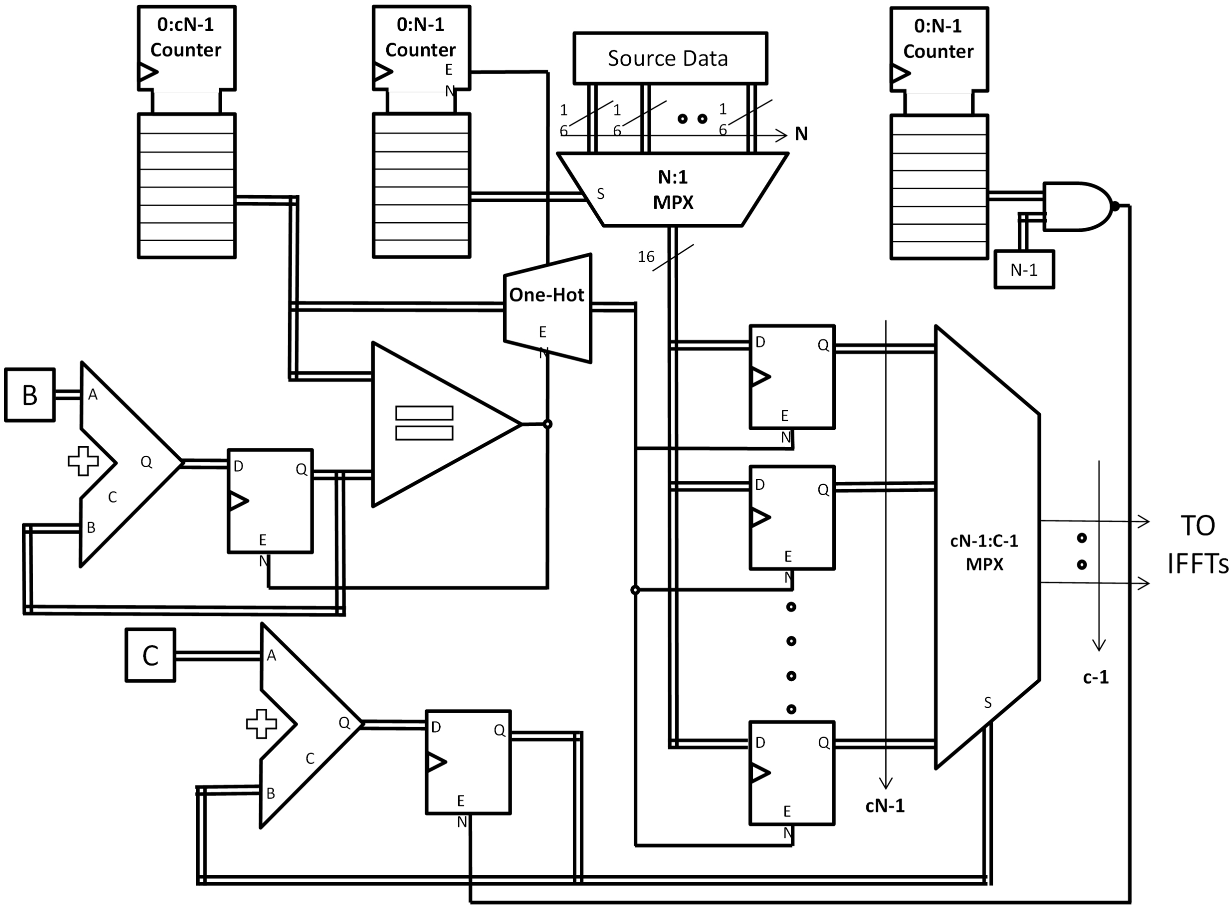

Figure 5.7: Logic diagram of Modulo Index circuit

In this alternative design, the divider used to point to the next valid input data value is replaced by a second counter. The purpose is to reference the next valid input data value determined by a 0 to $N-1$ counter, which is incremented every time a valid signal is present. This valid signal is provided from a counter of value 0 to $cN-1$; the output is compared to b. When a valid comparison is made, the value of b is added to the current comparison value so that the next valid comparisons will be $2b$, and so on. This process is repeated until all values up to $cN-1$ are exhausted.

5.2.2 Division versus Index Implementation Comparison

Table 5.1 illustrates the difference in resource utilisation and latency of the two methods described. For the table, $N = 16$, $b = 2$ and $c = 3$ resulting in $\alpha \approx 67\%$. The use of fixed-point data-paths and quantised coefficients leads to a finite Signal-to-Quantisation Noise Ratio (SQNR) at the output. Therefore, data-path precision must be dimensioned to achieve sufficient SQNR. A data-width representation of 16-bit is used, which satisfies an SQNR target of $\geq 60dB$. The target device is a Xilinx Virtex 5 XC5VSX50T.

Table 5.1: Divisor versus index method for modulo matrix augmentation

Metric	**Divisor**	**Index**	**% difference**
State Machine	3	2	-33%
RAM Blocks	2	2	0%
Multiplier	1	1	0%
Subtracters	1	1	0%
Adders	2	1	-50%
Counters	1	3	300%
Accumulators	0	1	100%
Comparators - $\geq$	2	2	0%
Comparators - $>$	8	8	0%
Comparators - $=$	0	1	100%
Comparators - $<$	3	3	0%
Comparators - $\neq$	0	1	100%
Multiplexors	1	1	0%
Total Flip-Flops	9697	9111	-6%
Total LUTs	19229	17224	-10%
Occupied Slices	6179	5811	-6%
Block RAM	18	16	-11%
DSP48E blocks	60	51	-15%
Max Estimated clock Period	$17.3ns$	$12.8ns$	-26%
Latency	$N.C.datawidth$	$N.C$	-93.7%

It can be seen that a modest reduction in logic utilisation of approximately 10%, results in a dramatically increased maximum clock frequency. This is due to the removal of the divider circuit load on the clock. The reduction in latency is due to the methodology itself; division in FPGAs is implemented as *long division*. This requires w clock cycles per division, where w is defined as the number of bits used to represent data symbols. The index method has no such divider and therefore offers the advantage of a 90% reduction in clock cycles required for an equivalent operation.

5.2.3 Phase Shift of IFFT Output Vectors

Data at the output of the IFFT are subject to a phase shift based on vectors derived from b and c. The required values are generated from complex exponential terms which are dependent on Sine and Cosine values. This approach is resource costly due to the need for Sine and Cosine coefficient generation methods such as CORDIC. An improved SEFDM architecture is found by using an IFFT to generate coefficient values, triggered when N, b or c are changed. Hitherto, the need for Sine and Cosine coefficient generation methods is removed. By recalling Equation (5.9), it can be re-written to reflect this improvement as:

$$x[k] = \frac{1}{\sqrt{N}} \sum_{i=0}^{c-1} \sum_{l=0}^{N-1} e^{\left(\frac{j2\pi i}{cN}\right)} \, s_b(i + lc) \, e^{\left(\frac{j2\pi lk}{N}\right)} \; 0 < k < N - 1 \tag{5.10}$$

where s_b is found by $s \; mod(b)$. Henceforth, the generation of the required complex exponentials is using such an IFFT or IDFT block. The source data is from a circularly shifting vector of length cN, for c iterations. This method introduces additional latency when b, c or N is changed. The latency is determined to be $cN \times c$ clock cycles. However, this methodology is valid for the purpose of creating a dynamic system for the evaluation of SEFDM signals.

5.2.4 The Effect of Sub-carrier Compression on FPGA Resources

In FPGA design the size of registers, memory or Digital Signal Processing (DSP) blocks must be explicitly stated at design time. Resulting logic may lay dormant until required and perhaps constitute a considerable portion of the total available resource. There is a trade-off between resource utilisation and design functionality. An increase in a maximum value of a design parameter can offer functional advantages, but at the expense of resource utilisation. For the SEFDM system, an increase in c leads to a corresponding increase in the number of N length IFFTs, complex multipliers and adder chains. Figure 5.8 quantifies the increase in resources required as c increases, as a measure of total FPGA logic utilised. This study shows that the maximum possible size of c, (here defined as $cMax$) when $N = 16$ is 12. Values of

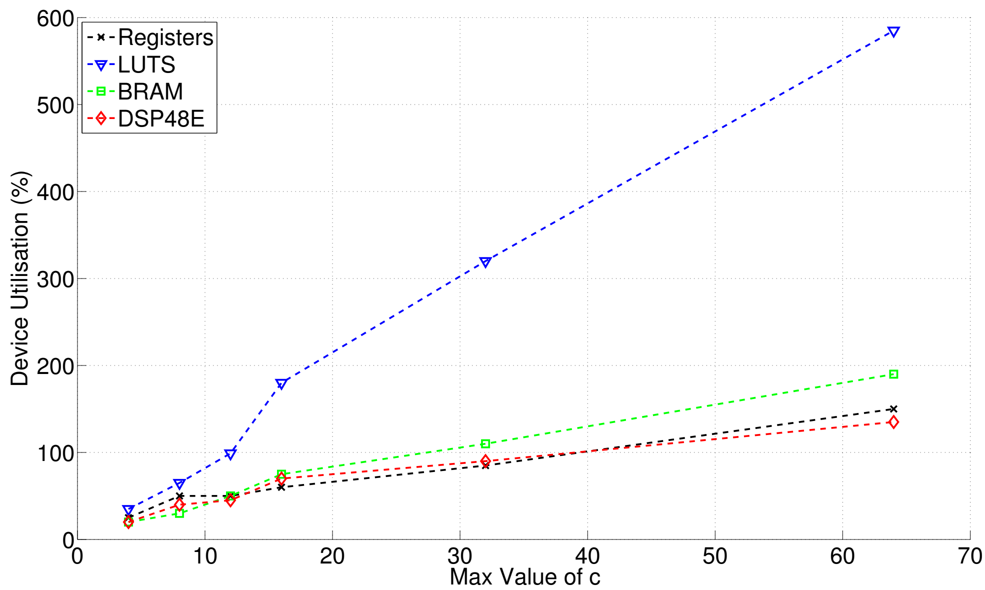

Figure 5.8: Chart showing the resource utilisation increase with respect to cMax

$cMax$ over 12 require more than 100% of the available resources. Thus, the value of 6 is chosen for cMax to illustrate the design, with a corresponding device utilisation of around 50%. For designs with a greater resource utilisation, the time penalty to synthesise from code to logic increases dramatically, hindering development.

The design architecture used to gather the data in Figure 5.8 reflects the lowest possible latency implementation via the use of parallel IFFT blocks. Conversely, the design also represents one with the most resource utilisation. A single IFFT block could be used in place of the parallel IFFT blocks. In this configuration resource is traded for system latency. The IFFT architecture employed here requires N clock cycles per transform. As a consequence a sequential IFFT method would offer an IFFT block reduction of approximately $c-1$, with an increased latency of $N \times (c-1)$ clock cycles. The resource saving is approximate, as an additional $cN:1$ multiplexer would be required and the latency penalty undesirable as data will output in bursts, introducing data gaps while IFFT processing takes place.

5.2.5 Signal Scaling

The input power and output power of a Fourier Transform are inherently equal. In the case of an IFFT used for OFDM generation, the output of the Fourier transform can be applied directly to a Digital to Analogue Converter (DAC). The Fourier Transform input values can be suitably large such that an output signal appropriate for the DAC ensues.

If smaller signals are required (such as those for SEFDM where post IFFT multiplication is performed), it may be tempting to reduce the input values to provide a correspondingly smaller output value. However, too small an input will artificially limit the output value range to only a small number of the possible output values. Most of the resulting output values will be less than one and contain fractional parts not representable using integers. Such values are rounded to the nearest integer and signals with defined curves (such as sinusoids) will appear heavily quantised.

A more suitable configuration is to have as large an input value applied to the IFFT as possible and scale the output values by some factor (usually a division or shift operation). This avoids output value rounding and provides a satisfactory signal representation. Unfortunately, when FPGA implementation is considered there is a resource penalty associated with a divider and a logical bit-shift operation is generally used in its place. However, bit-shifting is limited to an effective division by a power of 2.

Figure 5.9 identifies bit growth areas within the previously outlines method of SEFDM generation. The final output is bounded to within $\pm 2^{15}$ (i.e a signed 16-bit number) by the DAC. The largest possible positive or negative value is 32767, which can be derived by the multiplication of 128 and 256 (Powers of 2 are chosen so that shift operations can be employed to save resource over a divider). Due to an additional bit growth as a result of an adder before the final output, it is prudent to limit the IFFT output values and IDFT output values to 128. This results in a maximum

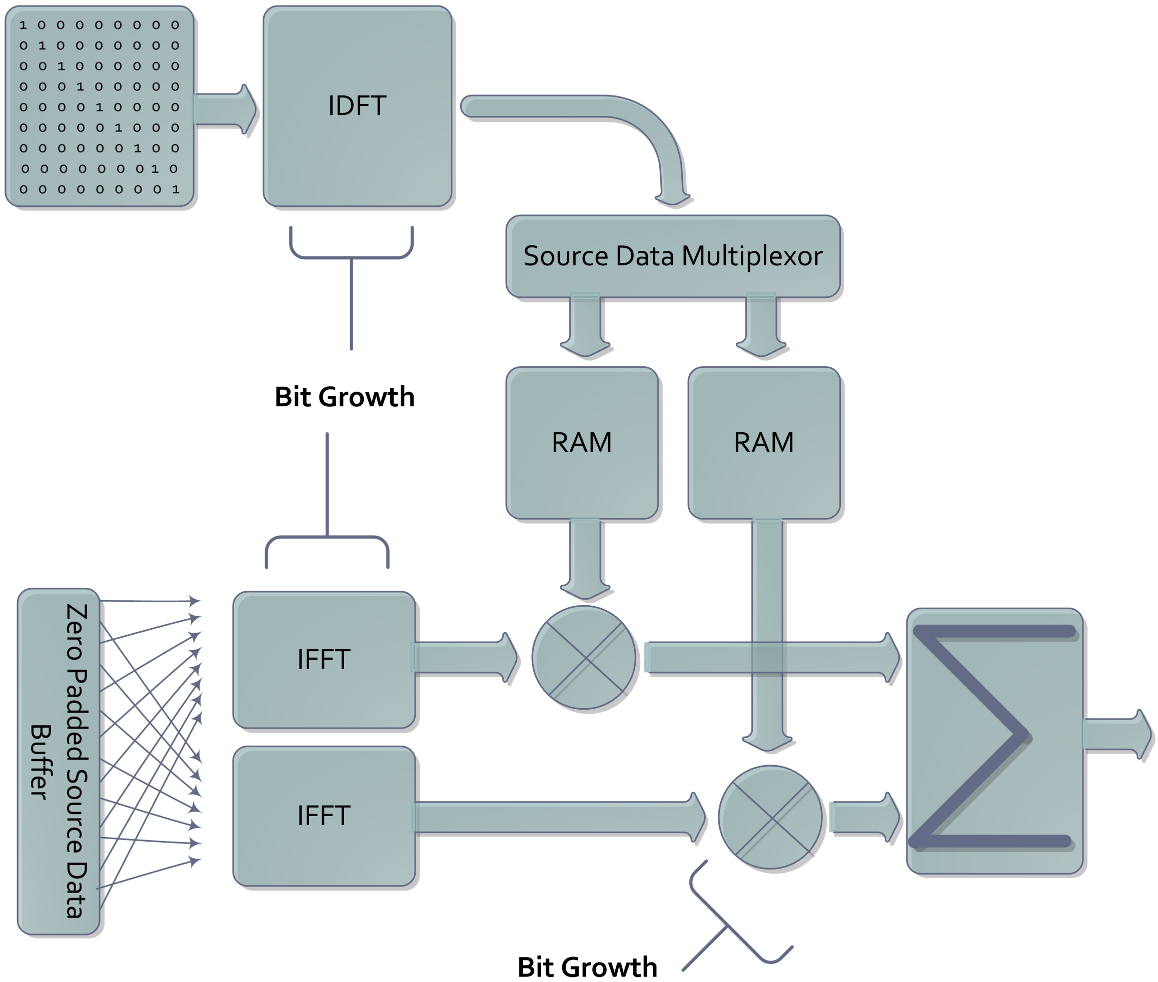

Figure 5.9: Figure to identify where scaling is required due to bit-growth in the SEFDM transmitter

post-multiplication value of 16768 and can accommodate value growth due to the effect of the output adder.

Formally, the largest possible output value (V_{Out}) when all input values (V_{in}) are at maximum, is defined as in Equation 5.11:

$$V_{Out} = V_{in} * N \tag{5.11}$$

where N is the number of IFFT points. If zero padding is applied to the input, the value of (V_{Out}) is instead as in Equation 5.12:

$$V_{Out} = V_{in} * N - Padding \tag{5.12}$$

Recalling Section 5.1.3, the size of the IDFT used to generate output shift vectors is not N, but cN. Importantly, the output values are derived from a limited input value (a single carrier) and therefore the output (V_{Out}) mirrors the input (V_{in}) as in Equation 5.13

$$V_{Out} = V_{in} \tag{5.13}$$

Depending on the values of c and N, the IDFT (of size cN) may not be a power of 2 and bit-shifting operations will provide too much or too little scaling. For example, if $c = 3$ and $cN = 48$ an input value of around 6000 and a scaling of 48 would provide a maximum output of 128. Due to the limitations of bit-scaling the actual maximum output value would be 192 (if 32 was the scaling factor) or 64 (if 64 was the scaling factor). It is preferable to use a larger scaling factor and increase the input value accordingly.

For preliminary testing, the scaling at the IFFT is set to be N, while the scaling at the IDFT stage is set to the next highest power of 2 above cN. Input values are set appropriately to produce the required maximum output of 128. In this configuration, simple bit-shifting can be used in place of a divider at the output of the IFFT. A study into optimum scaling and input values that provide the lowest error between theory and practical signals is performed in Chapter 6.

5.3 Functional Design Simulation

The system that was the subject of the previous section, is taken through simulated functional evaluation. Such a simulation is not based on a model of the expected operation, but instead is by the use of an application that interprets the design source code and evaluates it logically. In this configuration, the logical operations are completed identically to target hardware device. Such simulation is common for hardware development [20], [114], [115], [116] and is useful for inspection of logical operations as they are performed. This type of inspection is not possible once the design is loaded into an FPGA. The simulation uses a 32-bit Linear Feedback Shift Register (LFSR) to generate the input data values †. The software writes the values that are subsequently output by the simulation into a file. This file contains numerical values that correspond to analogue voltages. These values can be analysed and plotted to ensure correct operation of the FPGA simulation. Such a methodology is shown diagrammatically in Figure 5.10

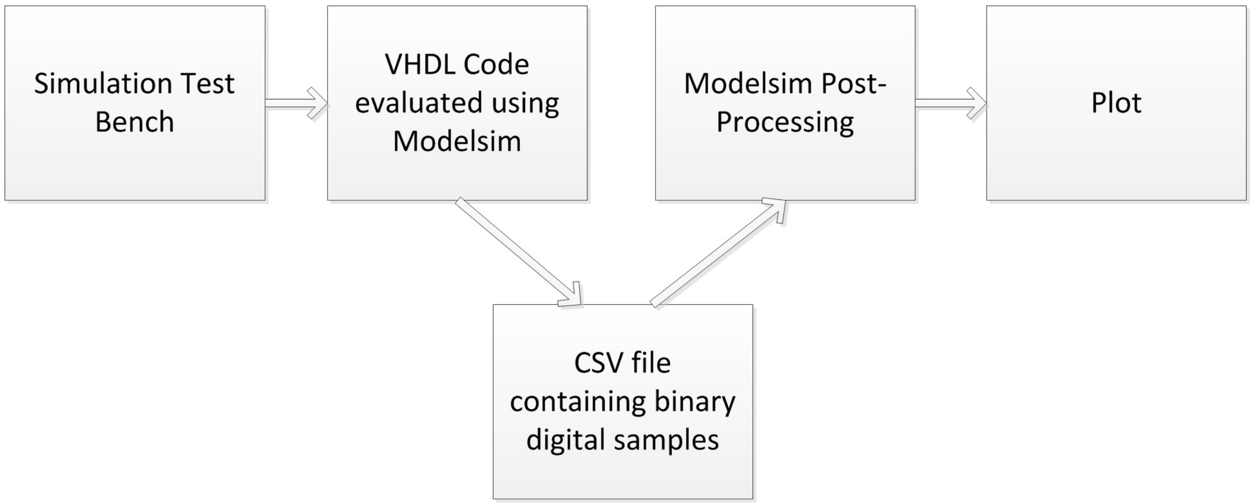

Figure 5.10: Overview of the method employed to enable Matlab processing of simulation generated results

Conceptually, a simulation test bench contains lists of input stimuli and a method to collect output values. This is a substitute for real data inputs from a user and real data outputs connected to a DAC. The test bench also provides board-level functions required for correct operation, such as clocking and resets. The test bench is used by the simulation tools to evaluate the logical operation of the code which describes the design. This is performed by the tools evaluating the underlying logical operations and interconnects, which the higher level language (VHDL in this case) is describing. The data captured at the output of the simulation is in a format suitable for a DAC and needs reformatting so that is can be displayed graphically. This includes

†Although not reported in this thesis, the author designed an M Sequence generator and implemented it in an FPGA at the start of this EngD work in 2007. The relevance of this design to the work detailed in the thesis is tangential. However, for completeness, the paper published describing the work [76] is reproduced in Appendix A

using the separate I and Q outputs to calculate the magnitude, which is subsequently plotted. The sample number forms the x-axis and the output magnitude the y-axis. This process is repeated with exactly the same parameters, except that the code is reconfigured for a different bandwidth compression factor (α) each time. Figure 5.11 shows the result of three system configurations, using the methodology previously detailed. Displayed is the output data for $\alpha = 1$ (OFDM), $\alpha = 0.5$ (Fast Orthogonal Frequency Division Multiplexing (FOFDM)) and $\alpha = \frac{2}{3}$ (SEFDM). $N = 16$ and zero padding is 4 which results in 8 data bits per SEFDM symbol. A total of 1024-bits of the output of the LFSR are utilised for each α plot.

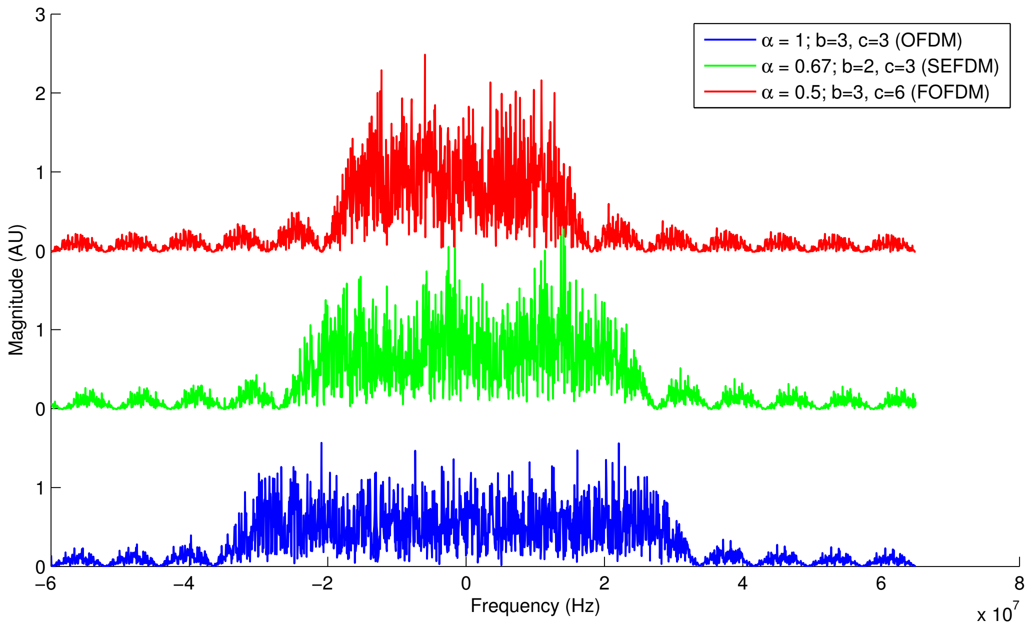

Figure 5.11: Power spectral plots for SEFDM with OFDM and FOFDM equivalent values of α

The spectral plots clearly show the correct operation of the design to generate SEFDM. However, it is not possible to judge if the signal content is correct. Therefore, a sample by sample comparison of the data described here, against analytical SEFDM models is performed in Chapter 6. This study provides definitive evidence of the correct operation of the design detailed in this Chapter.

5.4 Hardware Design Results

As previously detailed, the bounds that limit the flexibility and reconfigurability of the system must be set during design time. This section reports the resource utilisation with different design time parameters, such as c (used to calculate the bandwidth compression ration α) and N (the number of sub-carriers). Comparison against an equivalent implementation using DDS is performed. Such a method produces equivalent sub-carriers as those generated by SEFDM. Figure 5.12 reports FPGA resource utilisation based on the amount of LUTs consumed, for a range of N and c values. LUT utilisation is a metric commonly used to measure the size of an FPGA design [1].

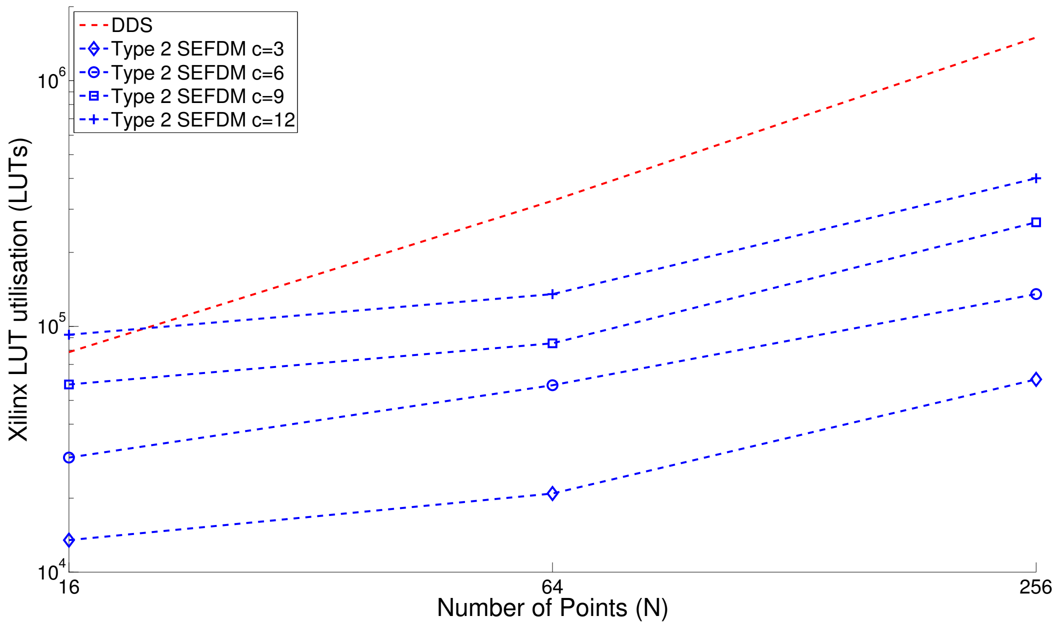

Figure 5.12: Plot showing actual FPGA resource utilisation for varying bandwidth reduction and sub-carrier values of SEFDM

From Figure 5.12 it can be seen that, as determined by estimation of complexity in Section 5.1.4, DDS requires a significantly higher number of operations compared to SEFDM to generate a signal with identical characteristics. Furthermore, the theoretical estimation of complexity is reflected in the FPGA resource utilisation with respect to N.

Additionally, the estimations of SEFDM complexity introduced in Section 5.1.4 is compared to reported resource utilisation. Figure 5.13 plots the estimated and reported FPGA LUT utilisation, for varying values of α and N. It is obvious that the theoretical complexity estimate and reported resource utilisation are closely correlated. This confirms the correctness of the complexity estimate model employed earlier and the overall superior performance of SEFDM compared to DDS with realistic bounds applied.

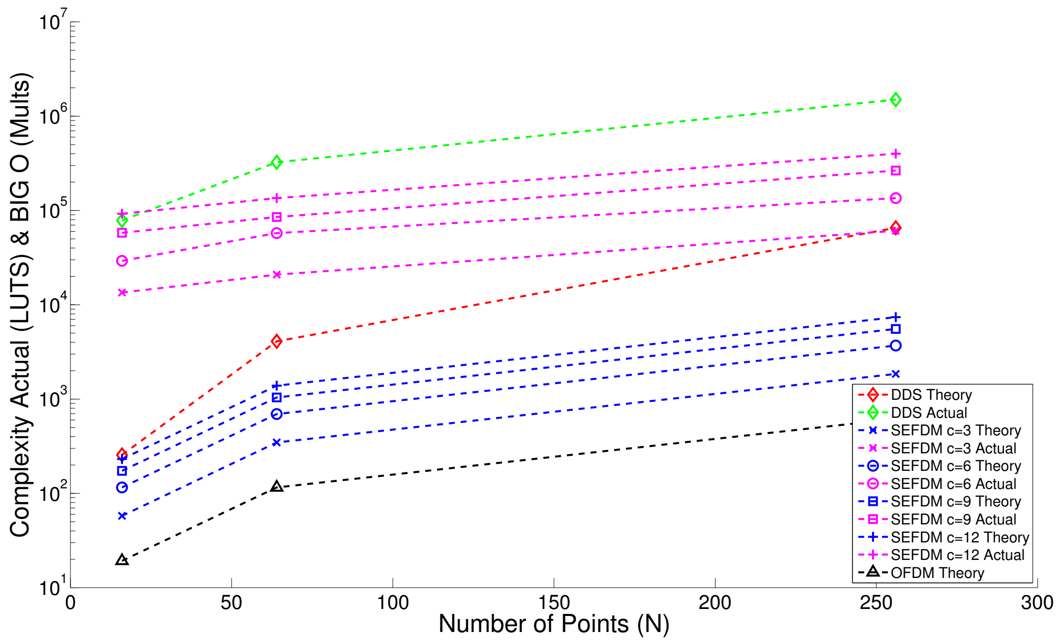

Figure 5.13: Plot showing actual and estimated resource utilisation for varying bandwidth reduction and sub-carrier values of SEFDM and DDS

5.5 Implementation Results

The design which was detailed in Section 5.2, together with the GUI detailed in Appendix D are evaluated together. This is completed by the setting of parameters in the GUI and by monitoring the analogue output of the system. Extensive testing was performed and the following represents a small subset to illustrate the correctness of operation.

The following represent a set of tests, where the systems response to a GUI instructed change in α is evaluated. Recalling that α represents the amount of bandwidth compression against standard OFDM. Therefore, a change in α should result in a reduction in the spectrum occupied at the output.

The output bandwidth is set to the maximum possible (i.e that of the DAC clock), with $N = 16$ and zero-padding $= 0$. All 16 carriers are therefore present in the resulting output. The data input is set to use an internal (to the FPGA) LFSR based Pseudo Random Binary Sequence (PRBS) of size 2^{16}. Figure 5.14 shows the GUI configuration used. Figures 5.15 to 5.19 show the change in the GUI parameters,

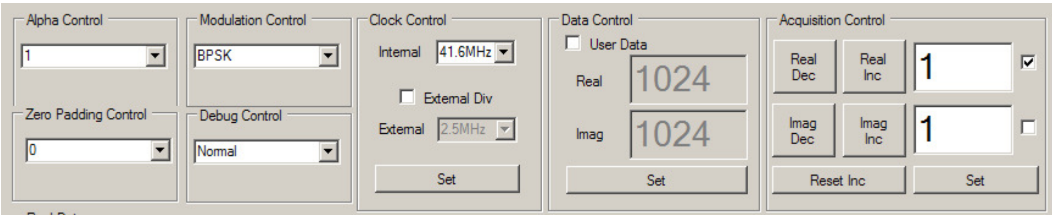

Figure 5.14: GUI settings used for the illustration of SEFDM generation

together with the resulting output spectrum.

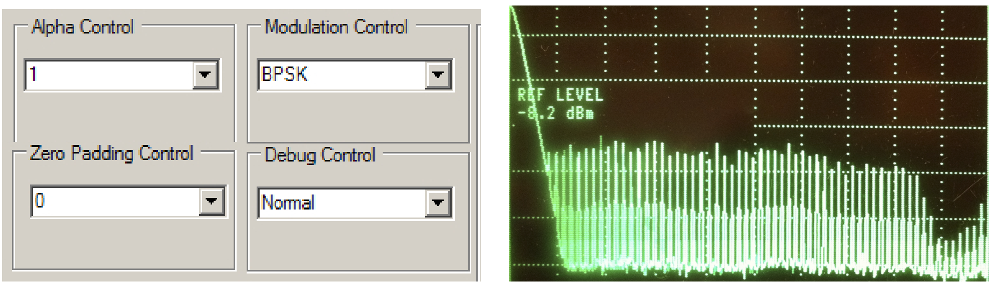

Figure 5.15: GUI setting and resulting spectrum for Alpha = 1

It can be seen that the bandwidth of the signal reduces in relation to a given α value set using the GUI. This confirms the correct end-to-end operation of the system as a user controllable real-time SEFDM generation platform. The plots however, do not conclusively determine that the resulting signal is correct. Therefore, Chapter

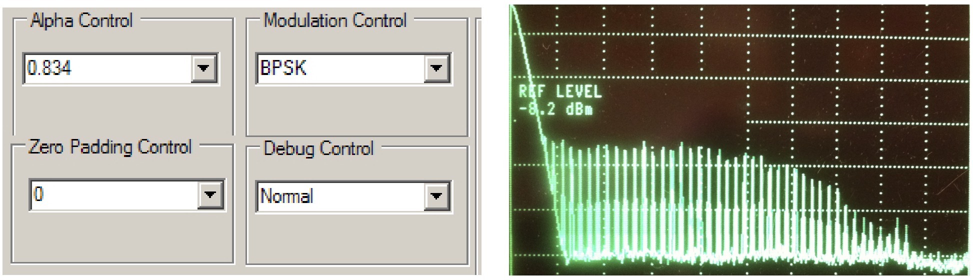

Figure 5.16: GUI setting and resulting spectrum for Alpha = 0.8

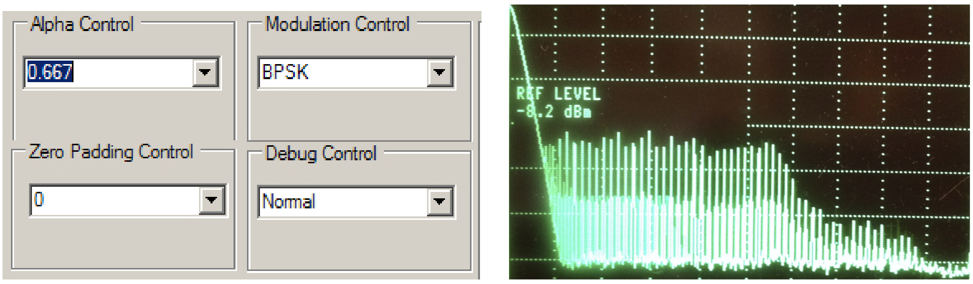

Figure 5.17: GUI setting and resulting spectrum for Alpha = 0.67

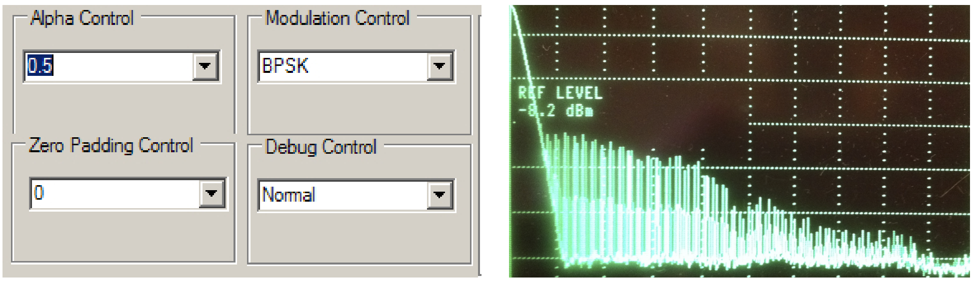

Figure 5.18: GUI setting and resulting spectrum for Alpha = 0.5

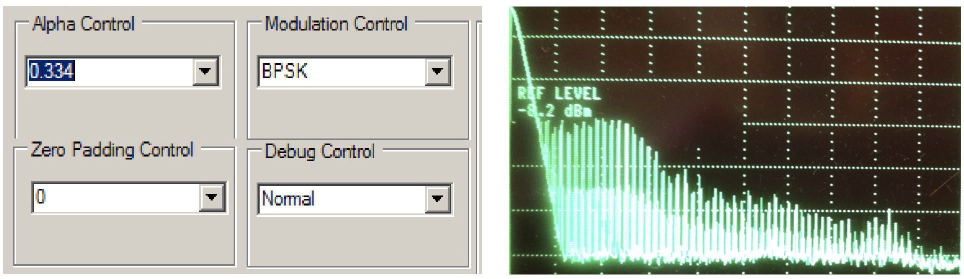

Figure 5.19: GUI setting and resulting spectrum for Alpha = 0.3

6 details an investigation into the signal validity. It includes comparisons to analytical signals and an attempt to use real signals, generated by the system, in an analytical SEFDM detector.

5.6 Critical Evaluation

In this section the limitations of the current design are discussed and where possible, suggestions are provided for improvement. The design itself is too large to have each element crucially examined at the lowest level. Instead, a holistic approach is taken where higher level functionality is appraised with low level design improvements suggested where appropriate.

One of the most limiting factors of the design is the IFFT which generates the SEFDM sub-carriers. By necessity, the use of an IFFT forces the use of a DFT to generate frequency shift vectors. The size of the IFFT is required to be a power of 2 and the DFT a multiple of 12. Hence, valid combinations of c and N must be selected to satisfy both IDFT and IFFT requirements. If N is a power of 2 (as is commonly implemented in standard wireless systems), the product of cN must provide a valid DFT size. For example, $c = 3$ and $N = 16$ produces a DFT size of 48 and $c = 6$ and $N = 16$ produces a DFT size of 96; both satisfy the DFT requirement. Conversely, $c = 2$ and $N = 16$ produces a DFT size of 32 which does not satisfy the DFT requirement. As c, along with b, defines the sub-carrier spacing some potentially attractive values (such as 0.75 as a product of $b = 3$ and $c = 4$) are not available. To remove this limitation, the use of pre-determined frequency shift values which are stored in RAM tables within the FPGA is required. This would enable any value of sub-carrier compression, within the pre-set bounds of the maximum value of c. A disadvantage would be the time and effort required to generate such tables and the additional design effort to move to a RAM based architecture.

Additionally, the use of IFFTs which require N clock cycles per transform, introduces a delay in the output of data. Output symbols appear in bursts rather than a continuous stream. One method of mitigating this effect is to have parallel IFFT blocks. In this configuration, one IFFT can process input symbols while the other IFFT is outputting processed symbols. By interleaving between the IFFT blocks a continuous stream of symbols is produced. This architecture would increase the resource requirement of the IFFT block, which may affect timing or other design elements. An alternative method is to clock the IFFT at double the clock rate of the input and output blocks. The output would be suitably buffered, so as to produce a continuous symbol stream. This approach does not scale easily (a doubling of the clock frequency would require very diligent logic placement) and would eventually necessitate the slowing of the input and output blocks.

Furthermore, SEFDM introduces a number of zero padded inputs to the IFFT block that do not need to be included in the IFFT computation. This method is exemplified in [113], where the concept of IFFT *Pruning* is introduced. Zero padded inputs are not included in the IFFT operation, reducing resource required within the FPGA. However, one of the principle motivations of SEFDM is its use of the standard IFFTs for the generation of sub-carriers. The use of a pruned IFFT necessitates customised transform blocks that are specific to SEFDM. This is both time-consuming and non-standard. Nonetheless, this is the subject of further investigation by Whatmough and others, to ascertain its feasibility when targeted for ASIC implementation.

5.7 Conclusions

This chapter demonstrates, for the first time, a flexible implementation of an FPGA based SEFDM transmitter. Values of two integers, b and c, control the spacing between the output sub-carriers. The values are limited only by the available resources present in a target FPGA device. The implications for different modulo arithmetic functions in VHDL are discussed and an optimum solution specific to SEFDM is presented. As an alternative to previous SEFDM generation methods, an additional IFFT block is advantageously included to produce a design suitable for FPGA implementation. The design is verified as functionally correct using simulation tools that evaluate the code of the design. The expected resource requirements based on design-time parameters have been analysed and maximum bounds have been discovered. These relate directly to the available sub-carrier spacing values.

Simulation results confirm that the operation performed by the code is correct. Comparisons are made of actual resource utilisation against theoretically derived values to validate the work. A critical examination identifies throughput and sub-carrier value limitations. Additionally, a method is introduced for reducing the required operations, for given values of b, c and N, by customising the IFFT block specifically for SEFDM. Such a method has been subsequently studied and targeted for use within ASICs. Results are provided of an end-to-end SEFDM system for different values of α. The architecture is specifically designed to be scalable as new FPGA devices become available. The maximum values of c or N can be increased with respect to the target FPGA resources, without any other changes to the code or to logic interconnects.

Chapter 6

Verification of a Hardware-Based SEFDM Transmitter

Publications relevant to this chapter are detailed as follows:

Perrett, M. and Darwazeh, I. *Verification of FPGA Generated SEFDM Signals*, London Communication Symposium (LCS'2011), London, UK.

Perrett, M., Grammenos, R. and Darwazeh, I. *A Verification Methodology for the Detection of Spectrally Efficient FDM Signals Generated using Reconfigurable Hardware*, IEEE International Conference on Communications (ICC'2012), Montreal, Canada.

Simulation, using a model of a system or process, is a widely accepted method of qualification and evaluation of complex systems. The success of a simulation depends on the accuracy of user defined values used to model parameters, which are gathered by analytical or empirical methods. In situations where some parameters are unknown or difficult to quantify, it is required to perform such operations in a real-world situation. Often, moving from a simulation environment to a hardware environment can result in considerable development and implementation challenges. Hence, simulation in conjunction with hardware, so called *in the loop* validation, can provide a fast method for verification and a reduction in development time.

This chapter concerns the successful transmission and reception of Spectrally Efficient Frequency Division Multiplexing (SEFDM) signals; the integrity and correctness of the transmitted signal is verified using in the loop methodology. Experimentally generated signals from a custom reconfigurable Field Programmable Gate Array (FPGA), previously introduced in Chapter 5, are sampled using an oscilloscope. Such data is formatted for detection using a model of the FPGA architecture, together with an analytical model of an SEFDM transmitter. Subsequently, such signals provide the input to an analytical receiver model, which is used to detect and confirm the experimental signals. In this configuration an SEFDM pseudo-transceiver is created, with practical signals detected successfully.

Implementation of multi-carrier wireless systems in FPGAs is commonplace [6] and [5]. Work on refinement of multi-carrier modulation methods, such as SEFDM [117], represents one of a number of emerging techniques which enable a reduction of required signal bandwidth. Such methods are without a corresponding penalty in data throughput and have a tolerable penalty in error rates [100]. Related techniques include Faster-than-Nyquist (FTN) [111], Overlapping Orthogonal Frequency Division Multiplexing (OvFDM) [24] and High-Compaction Multi-Carrier Modulation (HC-MCM) [23]. It has been shown that, by applying common measurement criteria such as Peak to Average Power Ratio (PAPR) [118], SEFDM has desirable properties for wireless communication. It has also been demonstrated that SEFDM transmitters can be successfully implemented in FPGA hardware [73] and subsequently targeted towards Application-Specific Integrated Circuit (ASIC) [74] [113].

Detector techniques required for SEFDM are well defined [119] [120] and specifically for the design in [121], are well suited for hardware implementation. Furthermore, preliminary work has been performed to target similar systems detection techniques in FPGAs [122]. However, other published detector logic architecture is both complex [33] and lacks clarity in important aspects; for example the method of implementation of matrix inversion and Cholesky decomposition. Therefore, this chapter serves as the first demonstration of a hybrid hardware and software environment, whereby experimental SEFDM signals generated using customised hardware [91], are decoded via an SEFDM receiver model which incorporates both Zero Forcing (ZF) and Sphere Decoder (SD) techniques.

Section 6.1 details the in the loop verification methodology, with a description of the variables used and a summary of the highest performing variable settings. Section 6.2 provides experimental signal verification against transmitter models. Furthermore, this section details the formatting applied to the experimental data to enable such comparisons and to allow compatibility with a detector model. Section 6.3 investigates the required number of time-samples and sampling rate(s) required at each stage of the verification process. Section 6.4 provides an introduction to the detector methods employed before error performance results, generated using practical signals, are provided in Section 6.5. The chapter is concluded in Section 6.6.

6.1 Verification Methodology

The system reported employs a hybrid hardware, test equipment and software topology. Hardware refers to an FPGA-based wireless signal generator with independent I and Q channels and a maximum output bandwidth of 125MHz per channel (Chapter 2 and [91]). The test equipment employed is a real-time oscilloscope (Lecroy Wavemaster 9300) capable of sampling rates from 10MS/s to 20GS/s. The software element is in two parts. The first part is a oscilloscope-sampled experimental signal from the FPGA hardware, which is phase aligned with an analytically generated signal to quantify the error between the experimental and analytical signals. The second part is an analytical model of the SEFDM receiver incorporating ZF and SD, which is used to detect the encoded input data. The verification process is shown diagrammatically in Figure 6.1.

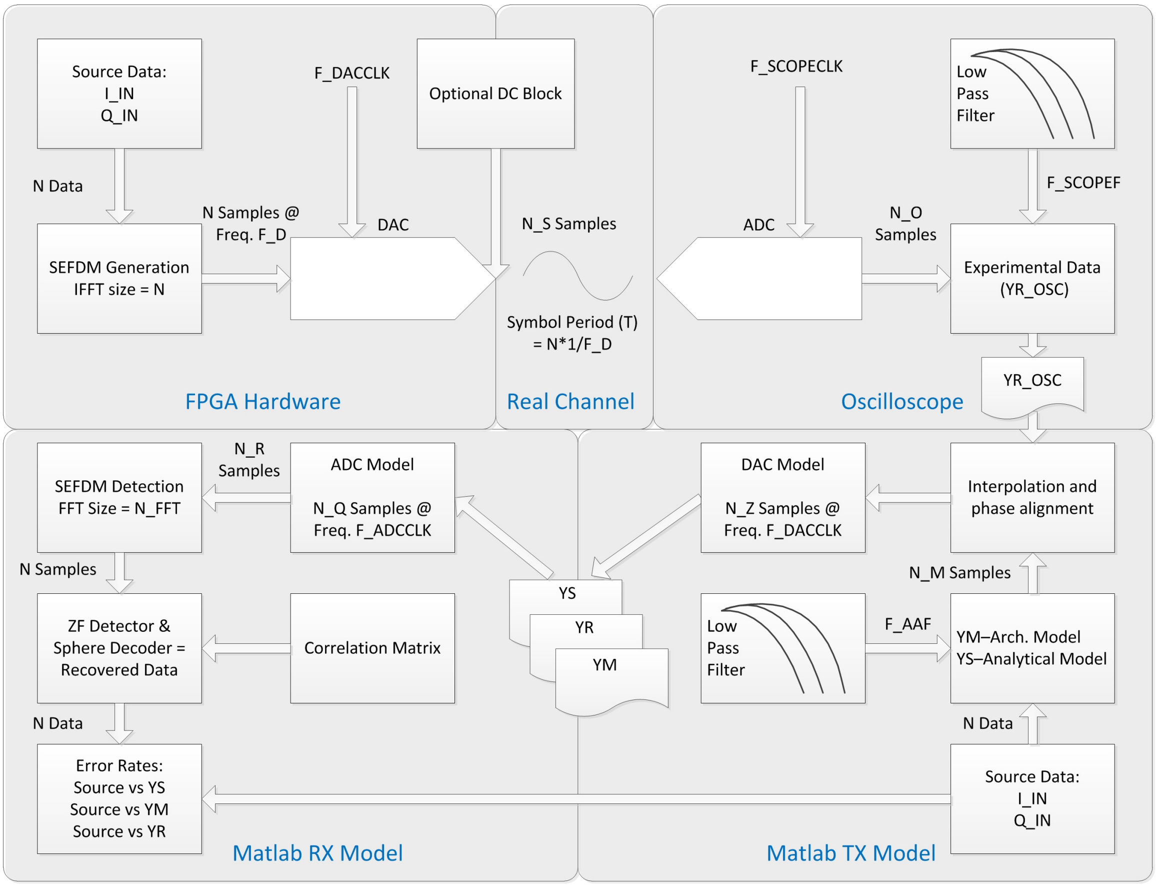

Figure 6.1: Verification methodology block diagram

6.1.1 General Description

Conceptually, the methodology is divided into five interconnected blocks, cross-referenced to the Figure in bold text. The **FPGA hardware** serves as the transmitter which generates analogue signals from numerically defined source data. This is followed by the **Real Channel** which is a physical connection (cable or wireless) from the Digital to Analogue Converter (DAC) in the FPGA block, to the input of the **Oscilloscope** (or equivalent). The oscilloscope is a real-time scope which outputs a digitised representation of an input signal at sample rate and filtering frequency set by the user. The **Matlab TX Model** comprises a set of mathematical functions which model the transmission process (both analytically and architecturally) as well as a mathematical representation of the transmitter data elements. These models are used for comparison to the sampled data fed to the **Matlab TX Model** as an ASCII file from the **Oscilloscope**. Finally, the **Matlab RX Model** comprises mathematical models of the detection process that include different detection algorithms, Analogue To Digital Converter (ADC) modelling and a specially designed mechanism for error analysis. This block performs the detection process for data generated from the hardware, as well as data generated from the architectural and analytical models. The hardware and measurement mechanisms are all designed to be reconfigurable for different signal formats and simulation parameters. It is not possible to evaluate large sequences of source data, due to the verification time required for each sequence. Therefore, as source data for the purposes of confirming the correct operation, a set of real-only source data sequences is selected at random from 256 possible sequences and are detailed in Table 6.1.

Table 6.1: Input data sequence selected

Sequence ID	**Data Value**							
1	+1	-1	+1	-1	+1	-1	+1	-1
2	+1	+1	-1	-1	+1	+1	-1	-1
3	+1	+1	+1	+1	-1	-1	-1	-1
4	+1	+1	+1	+1	+1	+1	+1	+1
5	-1	-1	-1	-1	-1	-1	-1	-1
6	+1	+1	+1	-1	-1	+1	+1	+1
7	+1	-1	-1	-1	+1	+1	-1	-1
8	+1	-1	-1	+1	+1	+1	-1	-1
9	+1	-1	+1	-1	+1	-1	+1	-1
10	-1	+1	-1	+1	-1	+1	-1	+1

FPGA Hardware

Complex source data (I_IN, Q_IN) of size N serves as the input to an FPGA based SEFDM transmitter, the design of which is detailed in Chapter 5. The FPGA presents to the DAC N time samples at data rate F_D. If F_D is lower than the DAC source clock F_DACCLK, the resulting analogue signal is constructed from N_S samples which is greater than N; otherwise N=N_S.

Real Channel and Oscilloscope

At the output of the FPGA system are analogue outputs that can be connected directly to an oscilloscope. Thus, even though the connection is via a channel the actual performance should be very close to optimum. The signal is sampled by the oscilloscope, at rate F_SCOPECLK, producing N_O samples (where N_O > N_S). This data can either be passed directly to the TX Model, or first be subjected to filtering at frequency F_SCOPEF.

Matlab TX Model

The same complex data applied to the FPGA (I_IN, Q_IN) serves as source data for two models; namely an analytical model of the complex SEFDM generation process [120] and a model of the FPGA architecture. The analytical model is based on the use of theoretically perfect (to Matlab precision of 128-bit floating point data representation) signal processing. On the other hand, the architectural model is designed to have identical limitations to the FPGA. The architectural model utilises a vendor supplied bit-accurate Matlab model of the Inverse Fast Fourier Transform (IFFT) used inside the FPGA. Furthermore, all data values are rounded to 16-bit integers.

The aforementioned N_O sample points and the resulting model(s) output data points (N_M) are made equal using interpolation. This is performed so that an accurate comparison can be made between the experimental signal and model signals. Furthermore, the experimental signal needs to be phase aligned to the analytical model, which is required as there is no phase correlation between the Matlab based signals and the experimental counterpart. The output of the TX models is three signals that should, in principle, be identical, as the procedure used to generate each of them is mathematically equivalent. The difference between each signal is identified by a cumulative absolute magnitude error process.

Matlab RX Model

As well as finding the error between each of the signals, an SEFDM receiver model is employed to detect the original input data (I_IN, Q_IN); forming a pseudo-tranceiver. Further sample rate changes are required to achieve this, resulting in N_Z number of samples for each data set. These are submitted to the receiver for each signal. The aforementioned N_Z samples have an ADC process applied, where the clock used (F_ADCCLK) is different from the transmitting F_DACCLK clock (as it would be in a real transceiver). The resulting samples of each model (N_Q) serve as input data to an SEFDM receiver model, based on the simple method defined in [100]. N_Q samples can be reduced to N_R samples, required for use with decoding techniques such as ZF and SD. The decoded data is then compared to the original source data and any error identified.

Optimum Parameter Settings

Based on empirical results, the variable settings detailed in Table 6.2 represent the highest performing. The derivation of variables F_ADCCLK, N_Z and N_FFT based on the variables F_DACCLK and F_SCOPECLK, is provided in Section 6.3.

Table 6.2: Highest performing variable settings used for the in the loop verification of SEFDM

Variable	Value	Unit	Description
F_D	250	MHz	Digital Data Frequency
F_BW	125	MHz	Analogue Signal Bandwidth
F_DACCLK	250	MHz	DAC Sampling Freq.
F_SCOPECLK	1000	MHz	Oscilloscope Sampling Freq.
F_ADCCLK	312	MHz	ADC Sampling Freq.
F_AAF	Off	N/A	Anti-Aliasing Filter Freq.
F_SCOPEF	Off	N/A	Oscilloscope Filter Freq.
N	32	Points	Transmitter Datapoints
N_O	256	Points	Oversampled Datapoints
N_M	32	Points	Model-only Datapoints
N_Q	80	Points	N_O Down-sampled Datapoints
N_Z	32	Points	Channel Datapoints
N_R	40	Points	Required SEFDM Datapoints
N_FFT	48	Points	SEFDM Receiver FFT Size

6.2 Transmitter Verification

As previously mentioned, the experimental signal needs phase alignment to the architectural and analytical models. The procedure used is a moving error calculation of the two signals; the lowest error is taken as the best case alignment. The sampling rate used to capture the experimental signal sets an upper bound on the phase adjustment that can be produced, defined in Equation 6.1 as in Equation 6.1

$$MaximumPhaseStep = \frac{1}{F\_SCOPECLK} \tag{6.1}$$

Setting F_SCOPECLK = F_DACCLK (this meets Nyquist as F_BW = F_DACCLK/2), would provide a 4nS phase adjustment step. Ideally, the phase alignment step size is required to be as small as possible, but in reality is constrained by current available devices. Hence, F_SCOPECLK is set as F_DACCLK*4 which results in an acceptable phase adjustment of 1ns and a corresponding 1GHz sampling clock.

The phase aligned experimental signal is compared to an analytical model signal and an FPGA architectural model signal for the purpose of error quantification. An example of such a comparison for the first data sequence in Table 6.1 is provided in Figure 6.2. It can be seen from Figure 6.2 that the three signals are roughly equivalent but that errors exist between the models and between either model and the experimental signal. It is expected that the experimental signal would have a error compared to the models. However, the error present in the experimental signal does not determine if such a signal can be detected using the previously outlined method incorporating SD.

Therefore, all of the experimental data sequences have a cumulative absolute magnitude error measurement performed against an equivalent signal generated, using the analytical model. For comparison, the architectural model error against the analytical model is also provided. This error (δ) is defined in Equation 6.2

$$\delta = \frac{\|S_{ref} - S'_{comp}\|}{\|S_{ref}\|} \tag{6.2}$$

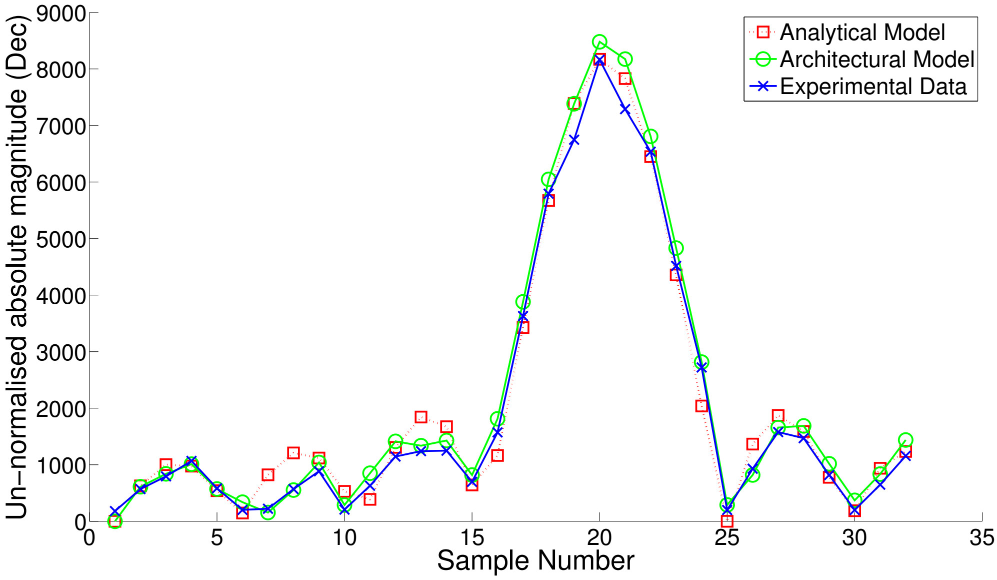

Figure 6.2: Comparison of experimental data compared to analytical and FPGA architectural models for sequence ID 1

where S_{ref} is the reference analytical model time-domain signal and S'_{comp} is either the experimental signal or the FPGA architectural model signal. The cumulative absolute magnitude error which equates to an average for each sequence (defined in Table 6.1) over all 32 samples, is illustrated in Figure 6.3.

Recalling from Chapter 5, scaling is applied inside the FPGA in order to reduce the signal magnitude to a value appropriate for the DACs; namely a 16-bit integer. There is no literature to determine the optimum setting and as such for the initial study a standard $\frac{1}{N}$ scaling is applied to each of the IFFT processes. However, it is easy to modify the architectural models scaling and therefore an additional two plots that reflect different scaling is provided.

The results in Figure 6.3 show that for the same input signal and the same bit scaling factor, the experimental and architectural signals have similar error trends and very close error values. The small difference in error values is attributed to the fact that the experimental system not only uses real electronic components but also a real transmission channel. Conclusively, the procedure used to generate the experimental

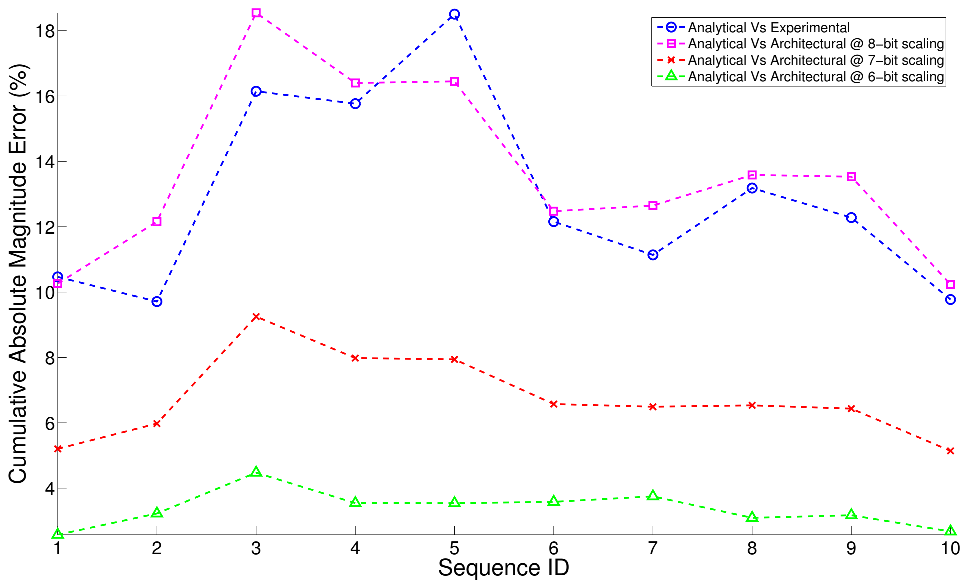

Figure 6.3: Cumulative absolute magnitude error for experimental and architectural signals with different bit scaling against analytical signals

signal is deemed correct. It can also be seen that a reduction in scaling applied (for example from 8-bit to 6-bit) results in a corresponding reduction in error. However, due to the fixed bit-width data representation of the DAC, scaling is necessary and therefore some small error will always be present between the model(s) signal and the measured signal.

Figure 6.4 provides the result of a study into the best performing bit scaling when applied to various input amplitude values and operational values (defined as *scale amplitude*), using the architectural model. The search is performed by the comparison of the resulting signal to a reference model-generated signal, with respect to a change in bit-scaling applied to the output of the IFFT. This process is then repeated for varying input magnitude values until all permutations have been evaluated.

The results in Figure 6.4 provide the optimum configuration of input amplitude and scale amplitude values for SEFDM generation. The optimum setting is found to be 2048 as the input amplitude combined with a 6-bit scaling factor applied to the out-

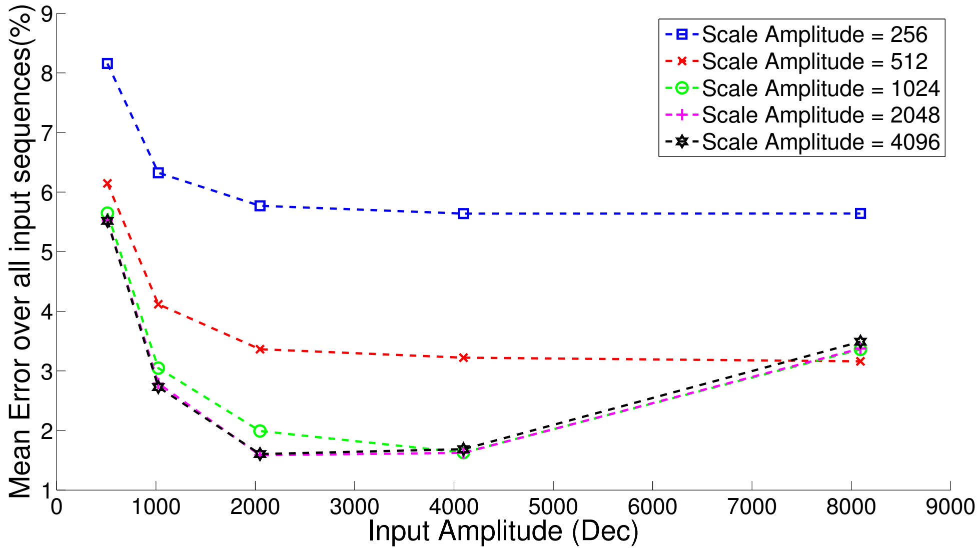

Figure 6.4: Error performance with different internal and external data amplitudes. Data representation in numerical decimal internal to the FPGA

put of the IFFT process. The lowest mean error, using the optimum values, between the analytical signal and architectural signal (and therefore the experimental signal) is 1.5%. This difference is accounted for by the use of fixed point arithmetic in the FPGA and the architectural model as opposed to floating point available using the analytical model.

The FPGA design is subsequently modified to generate new experimental signals based on the optimum scaling settings and the error comparisons completed over the 10 sequences repeated. Figure 6.5 Illustrates the error over the 10 sequences, between the new experimental signals and the original analytical signals. It can be seen that the error is now much lower (as expected) which confirms the validity of the architectural model to accurately mirror an FPGA generated signal. Such a model can in the future be used to generate much larger sequences, without time-consuming FPGA experimentation.

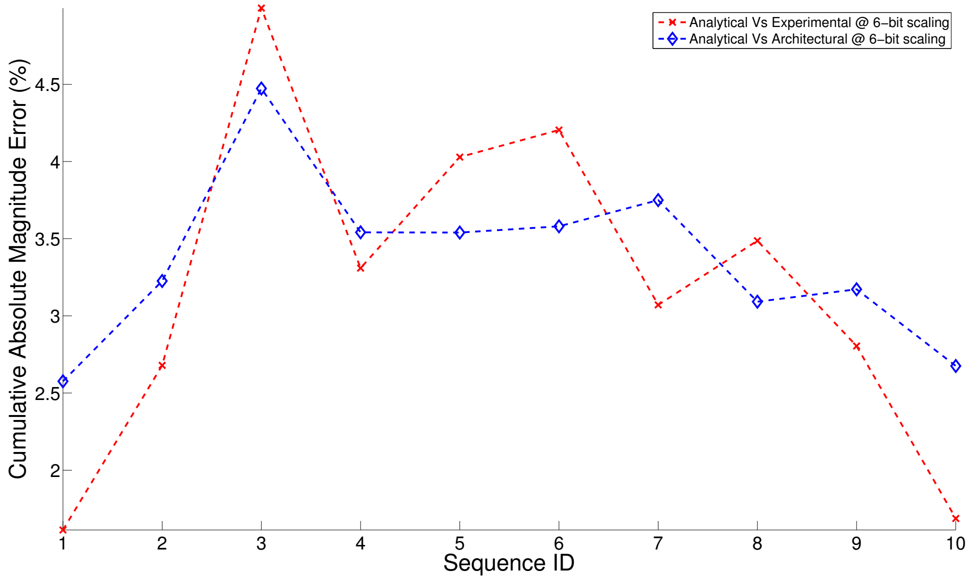

Figure 6.5: Error performance of new bit-scaling applied to the experimental data compared to architectural model estimate

6.3 Sampling Rate Discussion

The previously detailed DAC clock value, F_DACCLK, is fixed at 250MHz. Thus, every 4ns and on the rising edge of F_DACCLK, the digital input value presented to the DAC is translated to a proportional output voltage. The resulting analogue output signal has an effective maximum communications bandwidth, F_BW defined in Equation 6.3 as:

$$F\_BW = \frac{F\_DACCLK}{2} \tag{6.3}$$

Hence, the spectral properties of the data presented to the DAC cannot exceed F_BW without suffering aliasing effects. The resulting SEFDM symbol period (T) can also be defined, based on N and F_DACCLK, as in Equation 6.4. Due to the complex architecture utilised for the transmitter, N is both the number of time samples and the inverse transform size,

$$T = \frac{1}{F\_DACCLK} \times N \tag{6.4}$$

As F_D is set to be equal to F_DACCLK, the number of points which represents an SEFDM symbol in period T is N and hence, N_S=N. The SEFDM hardware transmitter is constrained in that both the inverse transform size and the number of time samples representing an SEFDM symbol (N) must be a power of 2; defined herein as N=32. For the detector the number of samples that each SEFDM symbol consists of must be related to N, but also consider the value of α, which is the bandwidth compression factor as defined in [123]. Hence, the transmitter and receiver require a different number of points to represent the same SEFDM symbol.

The minimum number of samples required for the forward transform at the receiver is found by $\frac{N}{\alpha}$, which can often result in a non-integer value. A search is thus undertaken to determine the appropriate forward transform size required to generate an integer number of time samples, defined as N_FFT and which must be larger then $\frac{N}{\alpha}$. Furthermore, this forward transform size must be a power of 2 (Fast Fourier Transform (FFT)) or a factor of 12 (Discrete Fourier Transform (DFT)) to suit available hardware. α itself is determined by two values, b and c, as $\alpha = \frac{b}{c}$. For example, taking b=5, c=6 and N=32:

$$\alpha = \frac{5}{6} = 0.834 \tag{6.5}$$

and the number of samples required for detection is:

$$\frac{N}{\alpha} = \frac{32}{0.834} = 38.4 \tag{6.6}$$

Taking the value from Equation 6.6 of 38.4, the next valid integer forward transform size which meets the aforementioned criteria is N_FFT=48. Due to the architectural differences between the complex transmitter and simple receiver, the number of input time samples required by the receiver (N_R), is not the same as the size of the forward transform (N_FFT) and is found using Equation 6.7.

$$N\_R = 48 \times \frac{b}{c} = 40 \tag{6.7}$$

The final detector architecture is therefore a 48-point forward transform, with the first 40 input data positions containing the sampled SEFDM symbol and with zero padding is applied to all positions over 40. The receiver and the transmitter will have different sample rates which are not solely based on the Nyquist criteria. Instead, the transmitting and receiving sampling rates have a relationship SR_R, as shown in Equation 6.8:

$$SR\_R = \frac{N\_R}{N} = 1.25 \tag{6.8}$$

Using the aforementioned F_BW value, it can be seen that with the Nyquist criteria, the detector sampling rate F_ADCCLK can be defined as in Equation 6.9 as:

$$\begin{aligned} F\_ADCCLK &= (2 \times SR\_R \times F\_BW) \\ &= (2 \times 1.25 \times 125 \times 10^6) \end{aligned} \tag{6.9}$$

F_ADCCLK, as calculated using Equation 6.9, is 312.5MHz. Therefore, as shown in Equation 6.10, the number of samples captured after the ADC (N_Q) is:

$$N\_Q = 2 \times N \times SR\_R = 80 \tag{6.10}$$

N_Q (=80) is larger than the required number of time samples found by Equation 6.7 (=40). Therefore, the captured signal is down-sampled to the required number of time samples. This is not attractive as interpolation processes are costly in terms of hardware resources. Hence, future investigation into the receiver architecture will concentrate on improved techniques to obtain a more accurate demodulation of the data.

For a simple SEFDM transceiver, the number of values taken from the forward transform is always equal to N, regardless of the transform size or α. Therefore, the original value of N used to generate the transmitted signal is restored regardless of the size of N_FFT.

6.4 Detector

The key problem with SEFDM is the ill-conditioning of the correlation matrix, which in itself is due to the internal Inter Carrier Interference (ICI) present due to the intentional violation of the sub-carriers orthogonality. Linear detectors, such as ZF and Minimum Mean Square Error (MMSE) are simple to implement but lead to significant Bit Error Rate (BER) degradation. ZF in particular, eliminates interference but significantly amplifies the noise present in the channel.

The detector employed in this paper is based on [100]. However, in this work the detector has been modified to include a binary search algorithm, which improves the likelihood of finding a valid solution at the expense of a slight increase in computational complexity. The original detector incorporates SD together with Real Value Decomposition (RVD), which doubles the dimensioning of the problem but at the benefit of allowing simple arithmetic operations when compared to the complex version. It has been shown in [100], that applying SD to SEFDM results in bandwidth gain at a significantly reduced computational complexity relative to Maximum Likelyhood (ML) and with a small error penalty. At the receiver, the input signal is initially demodulated using a DFT, whereby the outputs constitute the received statistics vector which when multiplied with the correlation matrix, give the ZF estimate. The correlation matrix contains numerical measures of all inter sub-carrier interference values.

The input to the SD includes the ZF estimate, the correlation matrix itself, as well as an initial sphere radius. The binary search algorithm employed in this work initialises the process with a very large radius and rapidly determines the minimum radius required to yield a successful solution. Having determined the values for the input parameters, the SD algorithm is initialised by carrying out an RVD of the received statistics vector and the correlation matrix. Since the ZF estimate has already been generated, the final step in the initialisation process involves the Cholesky Decomposition of the correlation matrix.

The main SD process is an iterative algorithm, traversing a tree until a complete path is found. On each iteration, a new estimate for the radius and the interval centre are provided with the aim of reducing the sphere size and converging to a solution, which provides the most accurate estimate of the transmitted symbols. The elements of the solution are determined by enumerating the possible lattice points of the sphere, which are determined by the current sphere radius. A reordering process known as Schnorr-Euchner (SE) [124], is also applied to enable the search to locate the lattice points nearest to the centre of the sphere. The detection process is shown diagrammatically in Figure 6.6 Furthermore, the procedure is broken down into the

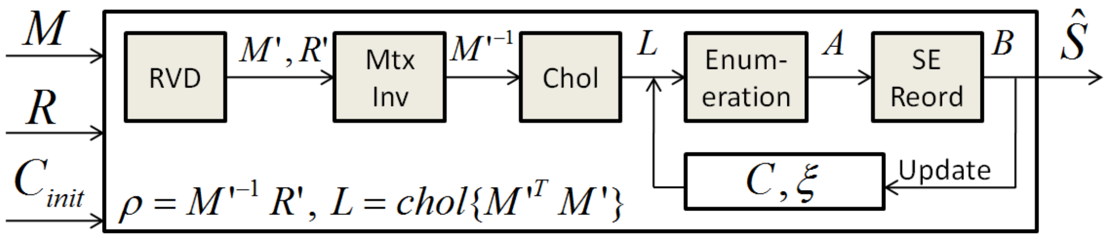

Figure 6.6: Diagram showing the Sphere Decoding (SD) process

steps below:

1. Re-sample the received signal to the number of points required based on the DFT size, N_R
2. Apply the DFT process to obtain the received statistics vector, R
3. Decompose the complex-values vector and matrix into a system employing real-valued numbers $M^{'}, R^{'}$
4. Compute the inverse of the de-composed correlation matrix, $M^{'-1}$
5. Multiply the statistics vector by the inverse of the correlation matrix as $\rho = M^{'-1}.R^{'}$
6. Apply Cholesky decomposition (L) to the correlation matrix
7. Execute the tree traversal iterative algorithm until a solution is found. The radius (C) and the interval centre (ξ) as well as the enumeration and the re-ordering processes are updated on each iteration

6.5 Results

The received experimental samples corresponding to the input sequences, detailed in Table 6.1 and processed using the methodology described in Section 6.1, are detected using the receiver detailed in Section 6.4. For each sequence, the analytical tool time taken before a complete tree path is found (termed here "time to success") and plotted against the error from Figure 6.3, the results of which are illustrated in Figure 6.7. Furthermore, Figure 6.8 shows the minimum radius required per sequence as a numerical measure of receiver complexity growth. As there is a defined time for each

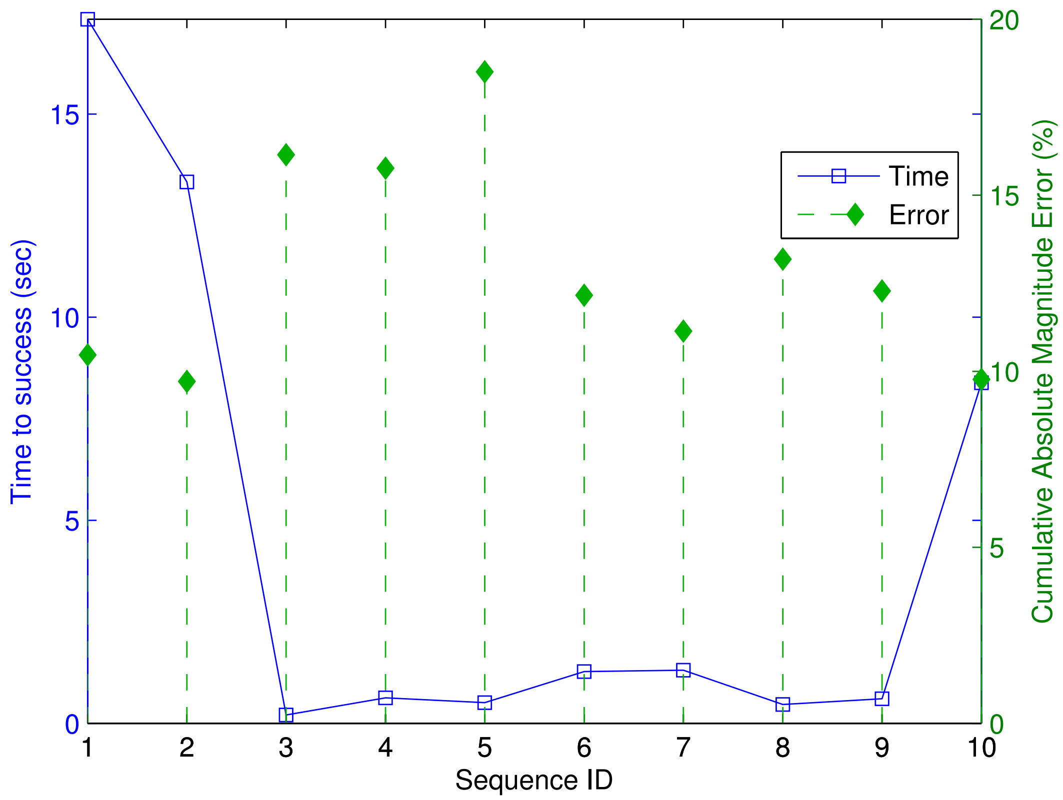

Figure 6.7: Detector results for data sequences in Table 6.1, quantified using analytical tool execution time

sequence the BER is zero and detection of the hardware-sourced data is successful. However, the time and complexity are subject to variation over data sequences. This difference should relate to the noise present in the signal to be decoded. It can also be seen that the error neither correlates with time nor complexity. This may be attributed to one or two reasons: either the quantification of the error is incorrect and underlying noise is not accounted for, or the decoder process is affected not just by noise but also by which sequence the input data belongs to. This can be confirmed by future analysis using the same technique as this work details, but with

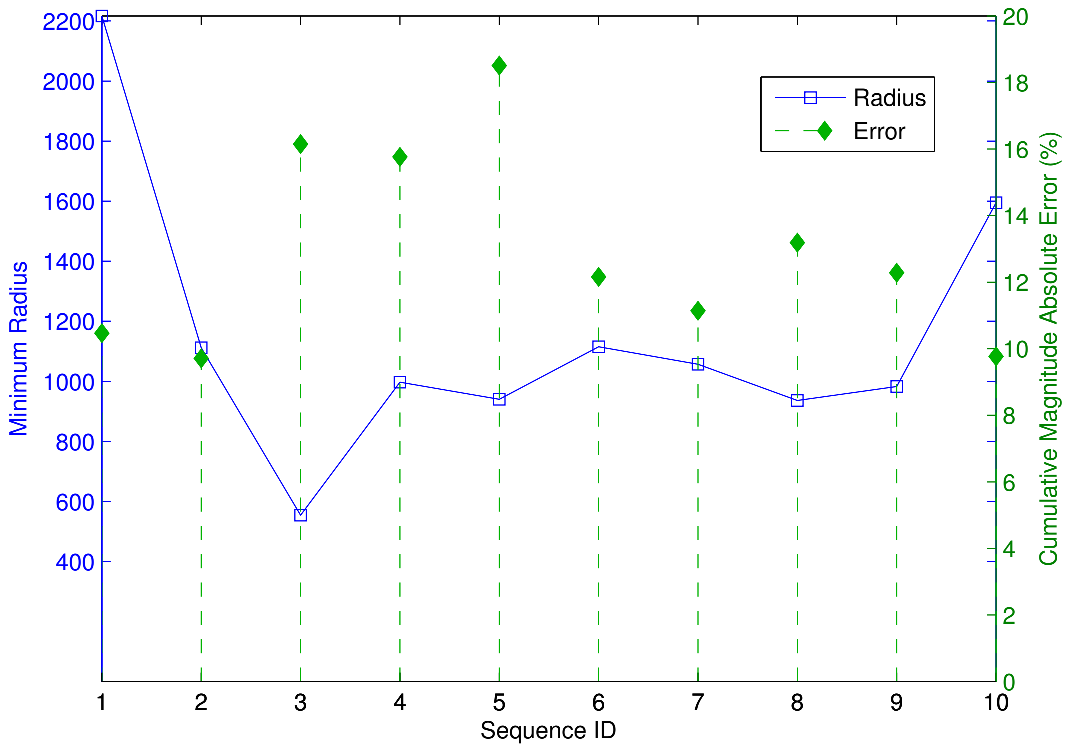

Figure 6.8: Detector results for data sequences in Table 6.1, quantified using minimum radius

larger sets of data sequences and channel models. The time to success is based on analytical tools, running in a simulation environment using an Intel core i7 Central Processing Unit (CPU). It has been shown in [31] that computation in hardware, either in a Digital Signal Processing (DSP) specific device or in an FPGA device, offers a significant increase in computational capacity per second. Therefore, it is expected that when hardware implementation of the receiver is realised, the time to success of the SD process will decrease significantly.

6.6 Conclusions

This work of this chapter verifies the correctness of the research and development approaches, described in previous chapters, concerning a customised and controllable FPGA-based signal generator for the purposes of generating SEFDM signals. It shows that SEFDM is adaptable for use in hardware and that such signals, allowing for errors due to the hardware architecture, can be successfully detected using previously reported SEFDM detection methods. This work also demonstrates the use of an *in the loop* methodology to enable the verification of the aforementioned experimental SEFDM signals, without a large SEFDM receiver development effort. Furthermore, such a verification gives credence for the need to encapsulate, in hardware, the analytical receiver model in order to realise an end to end SEFDM system.

Verification is also provided of the experimental signals, whereby the architectural model and experimental signals have similar errors when compared to the analytical model and serve to confirm the experimental signal properties are correct. Further analysis of the transmitter hardware architecture is provided to identify optimum scaling parameters internal to the FPGA. Such scaling is intrinsic to the current SEFDM transmission procedure due to hardware limitations. The optimum bit scaling and input amplitude values are identified resulting in an approximate error of 1.5% between the FPGA generated signal the analytical models. Such optimally generated signals are subject to sampling and serve as the input to an SEFDM detector model which successfully decodes the data. The feasibility of an SEFDM transceiver, where the signals are generated entirely in hardware, is confirmed and invokes further work on detector implementation in hardware to formulate a complete FPGA-based SEFDM end-to-end system.

Chapter 7

Conclusions

This thesis details the practical challenges of the implementation of spectrally efficient Frequency Division Multiplexing (FDM) signal generation. Standard hardware available was not sufficient to provide a platform with the required performance and interface capabilities. Therefore, Chapter 2 detailed the design, build, test and bring-up of a custom designed Field Programmable Gate Array (FPGA)-based platform, which also incorporated digital to analogue converters, memory and an Ethernet interface. The components are configured in such a manner to enable signals to be downloaded over Ethernet into memory, which are subsequently loaded into the FPGA where Digital Signal Processing (DSP) operations take place. The resulting signal is output to the inputs of four Digital to Analogue Converters (DACs), which in turn output analogue representations of their digital input. Additionally, the analogue signal can be up-converted using industry standard I/Q modulation techniques and an externally sourced local oscillator.

To ensure signals generated by the aforementioned system are correct, a verification procedure is designed for the hardware in Chapter 3. Using standard metrics of noise, transient response, frequency response and output linearity the performance of the system is characterised and confirmed to operate as designed. To enable communication using the aforementioned Ethernet interface, a custom FPGA-based stack is designed which enables simplistic communication; this is detailed in Chapter 4. This design is subject to theoretical optimisation and practical functional evaluation. Furthermore, comparisons are made to commercially available designs and considerable logic resource reduction is identified.

The implementation of Spectrally Efficient Frequency Division Multiplexing (SEFDM) using FPGAs is considered in Chapter 5. A comparative study of the chosen method of generation against a traditional bank of modulators using Direct Digital Synthesis (DDS) is undertaken and specifically targeted for FPGA implementation. Areas are highlighted where each method is attractive, using complexity estimates (Big $\mathcal{O}$ analysis). Furthermore, specific design considerations of the method chosen for implementation that relate to design-time resource constraints, bit-scaling and design optimisation are discussed. A final design is subject to simulated operation to confirm its correctness, before practical trials are performed using the FPGA. A custom Graphical User Interface (GUI) is developed (detailed in Appendix D), that enables dynamic reconfigurability of parameters within the FPGA; namely sub-carrier compression, modulation, zero padding and internal (to the FPGA) clock and control signals. Examples are provided of the correct operation in various configurations.

Standard signal demodulators are not capable of SEFDM signal recovery. Therefore, in order to qualify signals which have been generated using the previously described method, a hybrid verification method is detailed in Chapter 6. This method utilises a model of the expected FPGA output, which is compared to practical signals generated by the hardware and captured using a high speed oscilloscope. Furthermore, these signals are used with a model of an SEFDM detector, which successfully detects the practical source data. Further analysis is performed to compare the model of the FPGA hardware and the practical signal, which identifies improvements in the FPGA design. Subsequent hardware changes confirm the accuracy of the model by practical quantification. In future research on SEFDM, such a model can be used in place of an FPGA, which facilitates detector development and refinement, but maintains relevance to practical signal generation.

Importantly, this thesis represents a breadth of knowledge in complex system design, communication system implementation and the hybrid verification of signals. However, depth of knowledge is not provided in all areas and the reason is twofold. First, the thesis contains multiple projects which necessitated a wider than normal thesis scope. Second, the authors' company closed midway through the engineering doctorate period which limited information availability.

7.1 Thesis Achievements and Comments

The major achievements of this thesis are in furthering the knowledge of SEFDM signal generation. Specifically, the platform reported represents a milestone for wireless research (at the time of development) and enabled the implementation and generation of SEFDM without equal. The Ethernet interface, while not revolutionary, provided an elegant solution to a common problem. This design can also be reused in other designs which require similar functionality that is generally confined to expensive offload engines (so called TCP offload Engines (ToEs)).

The system developed is not without limitation, particularly the output bandwidth due to the addition of buffer amplifiers between the DAC output and the IQ modulator. Additionally, the time taken to produce a working SEFDM system has partially reduced novelty; much higher specification DAC and FPGA parts have recently become available (from mid 2011).

The Ethernet interface developed can be open-sourced, but this would require the development of a Media Access Controller (MAC) as the current design utilised vendor supplied IP blocks. Alternatively, an open-source MAC can be married to the design with some design effort. Due to time constraints, neither of these approaches were pursued.

The reported work on SEFDM generation targeted for FPGA implementation is, to the author's knowledge, a world first. By targeting the standard model from the literature to FPGAs, a new method is discovered which is suitable for hardware implementation. Furthermore, the use of the custom designed GUI provides a powerful method of generating SEFDM, by varying characteristics quickly and without the user's need to know the generation method. Future studies around SEFDM detection can leverage such a platform to generate practical SEFDM signals as required.

The design reported in the thesis is not claimed to result from an exhaustive search for the optimum architecture(s) to generate SEFDM. Instead, it is an attempt to optimise a single implementation, identify resource and latency improvements where the method of optimisation followed is at the logic block level, as opposed to gate or architecture level. Logical operations such as dividers and flip-flops are considered, but the gates (AND, OR etc.) that construct such blocks and the FPGA Look-Up Table (LUT) blocks such gates are translated to, are not considered. Ordinary design methods do not require such analysis as it is highly time-consuming. It is for these reasons that such an analysis is not attempted.

The use of the *in the loop* methodology quickly determines the validity of the practical SEFDM signals, without costly development into reciprocal hardware. Additionally, the model generated during this process is hardware accurate and can be used to generate source data for detector models in the future. Such detection models, when translated to hardware, can be used to form an SEFDM transceiver using the system developed in this thesis. The complexity of such an undertaking is reduced as prior knowledge of the characteristics of the practical signal are identified.

This method is successful for preliminary testing and verification of the practical SEFDM signals. However, the method of extracting the data from the oscilloscope is manual and could not be automated due to the available equipment limitations. Hence, large datasets (such as those required for Bit Error Rate (BER) analysis) cannot be utilised. Efforts to refine the architectural model to match the experimental signal characteristics partially mitigates this limitation. However, practical signal evaluation is still desirable, which is not possible using the design reported in this thesis. This limitation can be alleviated by either using a different oscilloscope or by the use of a reciprocal hardware (i.e an FPGA-based Analogue To Digital Converter (ADC)) to perform the signal capturing; the latter being attractive as such hardware may be used for detector implementation after verification has been performed.

7.2 Ongoing Related Work at UCL

The design and implementation of an SEFDM transmitter has been proven in hardware. Signals that such a system can generate are detectable using software models which conform to the literature. However, the detector used in this work is based on Sphere Decoder (SD) which does not suit hardware implementation, due to a variable complexity per symbol decoded [100]. Therefore, a hardware implementation should target a fixed complexity SD method, which has static complexity per symbol and is suitable for hardware implementation. Recent work on the implementation of fixed-SD, combined with Truncated Singular Value Decomposition (TSVD) methods based on [120], are viewed as a likely candidate for hardware SEFDM detection. It has been shown that TSVD, when combined with a sort-free strategy, has near fixed-SD performance and an architecture suitable for hardware implementation [125]. Furthermore, such an implementation has already been attempted and targeted for FPGA, with limited sub-carrier and modulation support [126]. Further work on moving to a hybrid FPGA and DSP based detector, with dynamic sub-carrier compression and modulation support is suggested. Beyond the work referenced here, the problem of converting current detector implementations from fixed to floating point is currently being addressed at UCL. This includes bit-width and truncation at each stage of the detector process and the effect this would have on performance.

7.3 Proposals for Future Work

The hardware platform would benefit from updating to utilise the current generation of devices. Xilinx and Altera now offer much larger FPGAs based on improved lithography processes. Furthermore, DACs such as the DAC34SH84 from Texas Instruments [37], offer 16-bit resolution and 1.5GHz sampling resulting in 600MHz analogue bandwidth. The verification performed on the system can be extended for more parameters, such as intermodulation, Total Harmonic Distortion (THD), in-band and out-band spurious signals. These parameters are not evaluated initially due to test equipment availability and due to the hardware platform not supporting such measurements. However, if the system is to be updated such measurements can be performed. Additionally, the FPGA stimulus code can be extended to provide suitable source signals for such measurements. As previously mentioned, the Ethernet interface does not have the necessary MAC element to allow for vendor agnostic FPGA implementation. Furthermore, development of this block would allow for a flexible approach to the integration of the stack. For example, removal of the buffers between the MAC and stack could be achieved. Additionally, removal of the transmitter state machine could be realised as the vendor data transmission requirements are removed.

The FPGA implementation of SEFDM can, in the author's opinion, be extended twofold. Firstly, the implementation can be extended to support a change in the value of N, which is the number of sub-carriers. This would enable a truly flexible and standardised platform, but will require further development effort and time. Secondly, by the continuation of the work of Whatmough et al [113], the current or future design can be implemented in an Application-Specific Integrated Circuit (ASIC) (current work is completed only in simulation). Such an undertaking would extend the resource bounds that the current implementation is limited by, as ASICs offer extended logic availability. There is a cost in completing an ASIC, so called *tape out*, which could be mitigated if SEFDM is included in future wireless standards.

Appendix A

M-Sequence Generator Paper

The paper reproduced here appeared in the 2010 Proceedings of London Communications Symposium. It reports work done by the author in the early stages of the EngD. The purpose of the work was for testing of audio systems, however, the design of the Pseudo Random Binary Sequence (PRBS) generator was used as the basis for the simulation of work in Chapter 5.

Implementation of a M-Sequence Pseudo Random Binary Sequence audio measurement system based on the Hadamard transform

Marcus Perrett, University College London

Abstract

This paper details the implementation of an industrial audio measurement system by the application of pseudo random binary sequences; where the system under test transfer function is computed using a Hadamard Transform. The procedure and benefits from this method are discussed and result in terms of computational complexity and on-line performance, are presented.

1 Introduction

In a manufacturing environment it is important to adhere to daily quotas to achieve profitability. When testing of a complex system is involved, as part of the manufacturing process, often a large initial outlay is required to build facilities which can provide both functionality and efficient manufacturing. Presented here is a computer based system which requires no test equipment and can therefore offer the lowest initial cost while performing required functionality by utilising advanced frequency characterisation methods.

2 Linear Time Invariant systems

The measurement of a system frequency response over its usable or specified bandwidth is known as the systems *frequency characteristics*. If a systems impulse response, $h(t)$ that is of Linear Time Invariant (LTI) type the convolution of its input $x(t)$ and its output $y(t)$ will contain the characteristics of the system, so long as the input $x(t)$ is suitably random. In discrete terms [1]

$$y[n] = \sum_{k=-\infty}^{\infty} x[n-k]h[k] \tag{1}$$

In terms of cross correlation, using (m) and (n) as time shifting variables:

$$\Phi_{xy}[m] = \mathbf{E}\, x[n]y[n+m] \tag{2}$$

where $\mathbf{E}$ is the expectation operator. Using Eq. (1) gives:

$$\phi_{xy}[m] = \sum_{k=-\infty}^{\infty} \phi_{xx}[m-k]h[k] \tag{3}$$

The result indicates that the input-output cross correlation is equal to the convolution of the input with the impulse response of the system. By utilising flat wideband characteristic signals as an input the relationship in time is thus [2]:

$$R_{yx}(\tau) = h(\tau) * R_{xx}(\tau) \tag{4}$$

where $R_{xx}(\tau)$ is the autocorrelation function of the input signal. The equivalent representation in the frequency domain is:

$$S_{yx}(f) = H(f).S_{xx}(f) \tag{5}$$

As $R_{xx}(\tau)$, the input signal, moves from deterministic to random its autocorrelation function approaches an impulse (δ(t)) function and its power spectrum characteristics $S_{xx}(f)$ becomes flat over the bandwidth. The cross correlation function therefore becomes equal to the system impulse response and the cross spectrum become equal to the frequency response of the channel. Signals which are random only over a specified period, so called pseudo random signals such as Pseudo Random Binary Sequences (PRBS), can also be used for this purpose [7]. The generation of these signals is discussed in the following section.

3 Binary Sequence generation techniques

A shift register with feedback, termed a Linear Feedback Shift Register (LFSR) is used to generate a PRBS signal, the two main types being Galois and Fibonacci [8]. The initial value in the registers is known as the *seed* and the positions of the feedback known as the *taps*. The taps determines the characteristics of the sequence, and the seed need only be non zero. A set of feedback Galois taps are written as:

$$[f_1, f_2, ...f_n]_g \tag{6}$$

where, f is the tap number and g is the Galois field. Although any tap position will result in a binary sequence output, some may repeat before the maximum length pattern and cannot be said to be random during the entire period. Only sequences with the maximum number of binary symbols before repeating can be called an M-Sequence. An LFSR can be represented as a polynomial of variable X [8], as shown in Eq. (7):

$$G(X) = g_m.X^m + g_{m-1}.X^{m-1} + g_{m-2}.X^{m-2} + .. + g_2.X^2 + g_1.X + g_0 \tag{7}$$

The coefficients g_n represent the tap weights, with '1' being connected and '0' being unconnected. The order m represents the number of stages. For example a 4-bit LFSR can have the form:

$$G(X) = X^4 + X^2 + X^1 \tag{8}$$

written using the Galois field representation:

$$[4, 2]_g \tag{9}$$

The method of determining if a particular $G(X)$ will produce an M-sequence is by determining if $G(X)$ is of primitive form and that it is a factor of $X^L + 1$, where L is the length determined by $2^N - 1$ and N the number of stages. Using Eq. (8), it can be seen that factorisation is possible as shown in Eq. (10) Therefore we can say that the tap positions in the example would not be suitable for M-Sequence generation.

$$X^4 + X^2 + X^1 = \pm(X^2 + X + 1)(X^2 - X + 1) \tag{10}$$

In general it is preferable to find all the tap positions for a given number of stages (N), rather than test each out individually. This can be achieved by first finding the Length (L) of the sequence before repetition, in this example $L = 15$ before finding the primitive factors of $X^{15} + 1$. The prime factors are:

$$(X + 1)(X^4 + X + 1)(X^4 + X^3 + 1)(X^4 + X^3 + X^2 + X + 1) \tag{11}$$

Primitive factors are the prime factors where the order is the same as the register size (N). The primitive factors are therefore:

$$(X^4 + X + 1)(X^4 + X^3 + 1) \tag{12}$$

It can be seen that the tap positions required to generate an M-sequence are one of the following:

$$[4, 1]_g, [4, 3]_g \tag{13}$$

4 The Fast Hadamard Transform

A method proposed by Hee [9], based on the work of and Cohn and Lempel [11], and Chu [10], uses a Hadamard Matrix and the Fast Hadamard Transform (FHT) [12] to produce the impulse response of a system when an M-Sequence is applied. The FHT is a reduced complexity variant the Hadamard Transform which is itself based around the Walsh Series [13]. Similar to an FFT, the FHT requires only $N.Log(N)$ operations. For a given M-sequence a matrix can be constructed using a sliding window across a hypothetical output sequence. A 7-symbol length M-Sequence would generate a 7x7 matrix as shown in Eq. (14)

$$\begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \\ h_5 \\ h_6 \\ h_7 \end{bmatrix} = \frac{1}{7} \begin{bmatrix} -1 & -1 & -1 & +1 & +1 & -1 & +1 \\ -1 & -1 & +1 & +1 & -1 & +1 & -1 \\ +1 & +1 & +1 & -1 & +1 & -1 & -1 \\ +1 & +1 & -1 & +1 & -1 & -1 & -1 \\ -1 & -1 & +1 & -1 & -1 & -1 & +1 \\ -1 & +1 & -1 & -1 & -1 & +1 & +1 \\ +1 & -1 & -1 & -1 & +1 & +1 & -1 \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_5 \\ s_6 \\ s_7 \end{bmatrix} \tag{14}$$

Which can be summarised as:

$$[h] = \frac{1}{7}[M_7][S_0] \tag{15}$$

$[M_7]$ is the left circularly delayed version of the original M-sequence with a period of seven times the bit period. $[S_o]$ is the output sequence and $[h]$ the cross-correlation. It can be seen that the matrix is recursive in both rows and columns. Since $[M_7]$ only contains ± 1, the multiplication of $[M_7][S_o]$ requires only addition and subtraction operations. It has been noted by Lempel [11] that the cross-correlation matrix is similar to a Hadamard matrix. If the required permutation matrices are constructed and the Cross-Correlation Matrix are re-ordered to a Hadamard Matrix, then it allows the use of the FHT. In order to realise this translation, the matrix in Eq. (14) must be decomposed into a L x N and a N x L matrix, here called α and β. β is the inverse of the first N rows of the matrix $[M]$, α is found by:

$$\alpha = \beta' \sigma^{-1} \tag{16}$$

β' is the Prime Transposed Matrix of β and σ^{-1} is the Matrix Inverse of an N x N matrix of β. Using these 'tags' the resulting cross-correlation matrix from the Device Under Test (DUT) can be converted to a format suitable for use with the FHT.

5 Implementation

The method of obtaining an impulse response using the Hadamard Matrix equivalent is thus:

1. Generate an M-Sequence and compute the α and β 'tags'.
2. Modify the M-Sequence DC offset, $0 \rightarrow 1$, $1 \rightarrow -1$, and apply to the Device Under Test.
3. Capture the resulting M-Sequence and re-order according to β tags.
4. Apply a FHT to the β permuted data. The output will be sequency ordered.
5. Re-order the post FHT data according to the tags of α.
6. Perform a FHT on the data. The systems frequency response has been realised.

Such an implementation lends itself to being implemented in a programming language such as C++, due to its direct memory access properties and efficient maths functions.

5.1 System details

The system purpose is to evaluate the frequency characteristics of an automotive audio amplifier. It is designed around a standard PC, with a PCI I/O card controlling a custom switch box to route signals from the 15 outputs of the DUT to a single line-in connector of a sound card which is used here as an ADC. The signals are in the audio frequency range with a bandwidth range of 20Hz to 22KHz. Other interfaces included but not discussed here are Common Automotive Network (CAN) used to control the DUT and UART over USB to read-back messages from the DUT. The block diagram shows the interconnections in Figure 1.

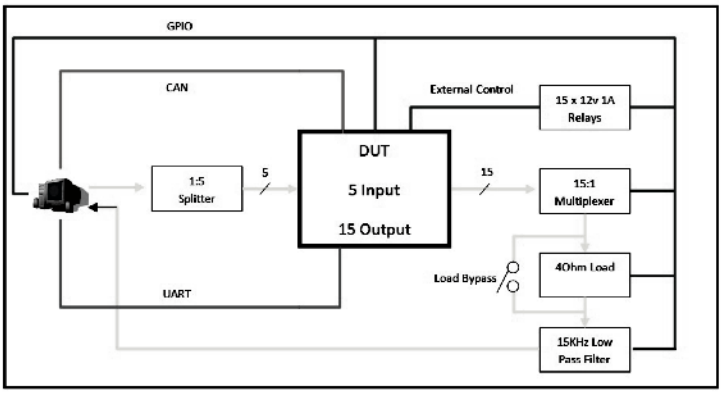

Figure 1: Hardware system block diagram

5.2 Performance Results

In terms of computational efficiency compared to other methods the FHT was compared with traditional cross-correlation (fast methods such as [14] and [15] exist but are not evaluated here). The values for Cross-Correlation were taken from [14]

	Addition/Subtraction		**Multiplication**	
Method	**Numerator**	**Denominator**	**Numerator**	**Denominator**
Cross Correlation	$N(M-N+1)$	$2N(M-N+1)$	$N(M-N+1)$	$N(M-N+1)$
Hadamard	$N\log(N)$		0	0

Table 1: Computational analysis of cross correlation and the FHT

It can be seen that the Hadamard transform offers superior computation performance compared to the cross-correlation method. It is important to note that the 'tags' of any N x N matrix must be computed once for any new value of N, and that Table 1 does not include the data permutation as these do not require any operators and can be implemented with little computation time penalty. The system was evaluated through measurements of the DUT assessed in term of line throughput against a quotation specification of 30 tested devices per day (where a day here is defined as $T_D = 510$ working minutes at 80% efficiency). Each device has 15 outputs (N_O) and 8 inputs (N_I) and required to be fully tested twice. The time per measurement allowed was thus:

$$M_T = \frac{T_D}{(N_O.N_I.2)} \approx 120s \tag{17}$$

A measurement of the time to perform a single measurement on average was under 52s.

6 Conclusion

This paper concerns a method for generating Pseudo Random Binary Signals and resolving a systems frequency response by application of the Fast Hadamard Transform. The aim is to produce a system utilising this method for testing commercial audio systems. In the paper the Pseudo Random Binary Sequence and Maximum Length Sequence generation is discussed and used to introduce the concept of the Fast Hadamard Transform. This is implemented as an alternative to traditional cross-correlation techniques in conjunction with the generation of a permutation matrix for translation into the format required for the Fast Hadamard Transform. Details of the system construction and connection are provided, along with estimated computational complexity and real timing of the system performance in a manufacturing environment. This system now in use in both the UK and China for manufacturing line testing of the target product.

References

[1] Wolfgang Fuerst Paul A Lynn. *Introductory Digital Signal Processing with Computer Applications.* John Wiley and Sons, second edition edition, 2007.

[2] Zaiton Sharif and Ahmad Zuri Shaíameri. The application of cross correlation technique for estimating impulse response and frequency response of wireless communication channel. *The 5th Student Conference on Research and Development ñSCOReD*, 2007.

[3] Naoya Moriya and Yutaka Kaneday. Impulse response measurement that maximizes signal-to-noise ratio against ambient noise. *Acoustical Letters*, 28, 2007.

[4] S. Wang Q. Meng, D. Sen and L. Hayes. Impulse response measurement with sine sweeps and amplitude modulation schemes. 2008.

[5] Angelo Farina. Simultaneous measurement of impulse response and distortion with a swept sine technique. Paris,France, 2000. Presented at the 108th AES Convention.

[6] ARCHAMBEAU Dominique STAN Guy-Bart, EMBRECHTS Jean-Jacques. Comparison of different impulse response measurement techniques. *Sound and Image Department, University of Liege*, 2002.

[7] Ole Herman Bjor. Maximum length sequence. www.norsonic.com.

[8] Andrew M. Klapper Mark Goresky. Fibonacci and galois representations of feedback-with-carry shift registers. *IEEE Transactions on Information Theory*, 48, 2002.

[9] J. Hee. Impulse response measurement using mls. pages 193 – 205, 1990.

[10] W Chu. Impulse response and reverberation-decay measurements made by using a periodic pseudo random sequence. pages 135 – 137, 1977.

[11] Martin Cohn and Abraham Lempel. On fast m-sequence transform. *IT-23*, 23, 2003.

[12] Henry Y. L. Mar and C. L. Sheng. Fast hadamard transform using the h diagram. *IEEE Transactions On Computers,*, October 1973.

[13] P.M Ralmondo Tallia. The walsh hadamard transform: An alternative means of obtaining phase and amplitude maps. 1984.

[14] Jae-Chern Yoo and Tae Hee Han. Fast normalized cross-correlation. *Circuits Syst Signal Process*, 28:819–843, 2009.

[15] Shyamalee Mukherji. Fast algorithms for binary cross-correlation. *Geoscience and Remote Sensing Symposium*, 1, 2005.

Appendix B

PCB Schematic Files

This appendix contains the schematic diagrams for the Field Programmable Gate Array (FPGA)-based wireless signal generator detailed in Chapter 2. The structure of the schematic is as follows. Figure B.1 illustrates the top level component connectivity. Figure B.2 and Figure B.3 detail the Digital to Analogue Converter (DAC) and IQ modulator connections and component configuration respectively. Additionally, Figure B.4 shows the DAC-specific power and decoupling components.

Figure B.5 contains Ethernet connectivity and component connections while Figure B.6 and Figure B.7 detail the FPGA; the former contains the FPGA per bank connections and the latter the flash and Joint Test Action Group (JTAG) connections. Figures B.8 and B.9 detail the Double Data Rate (DDR) memory connections for all four devices, two per page.

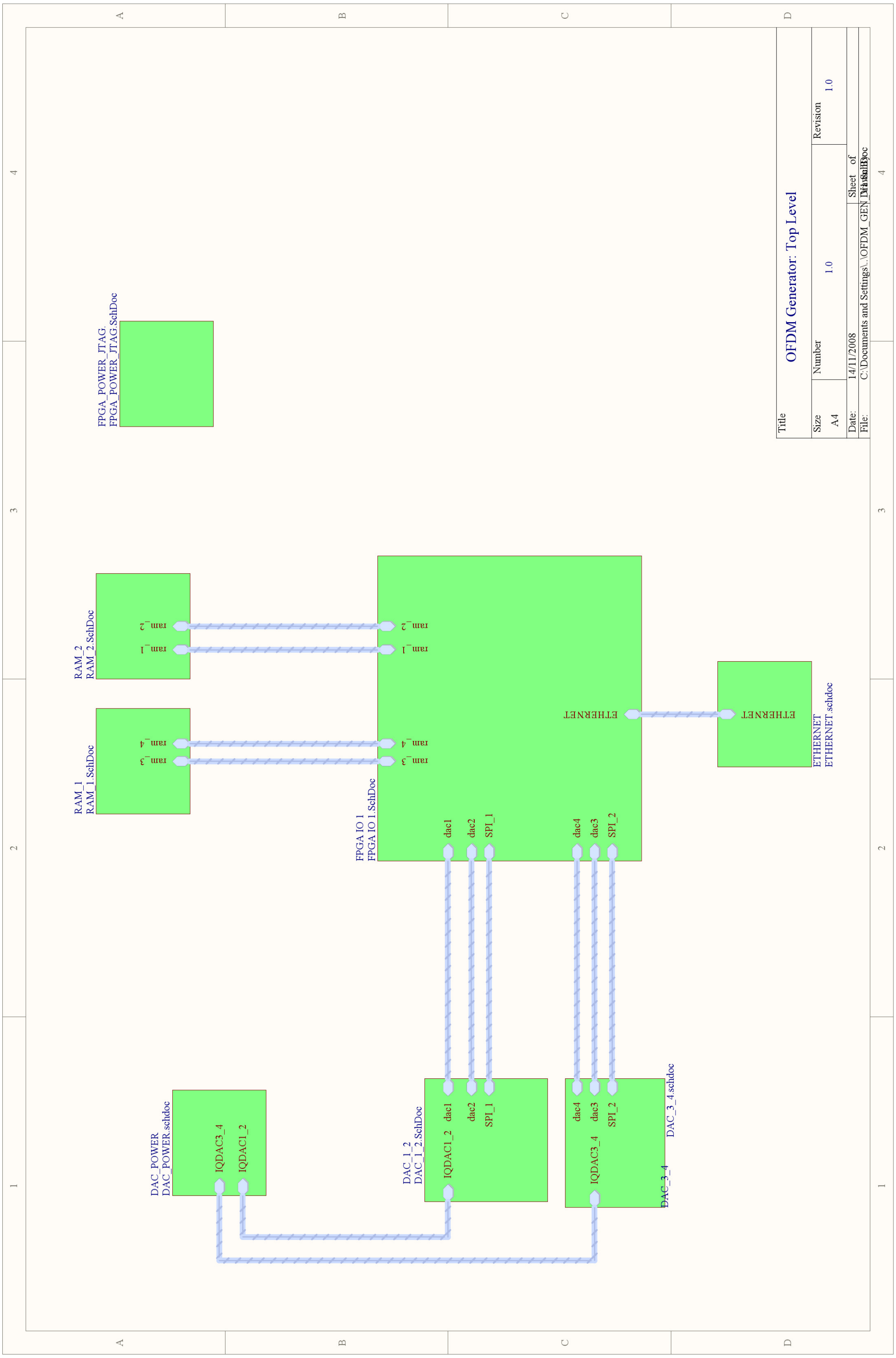

Figure B.1: Schematic page 1 - Top level component connections

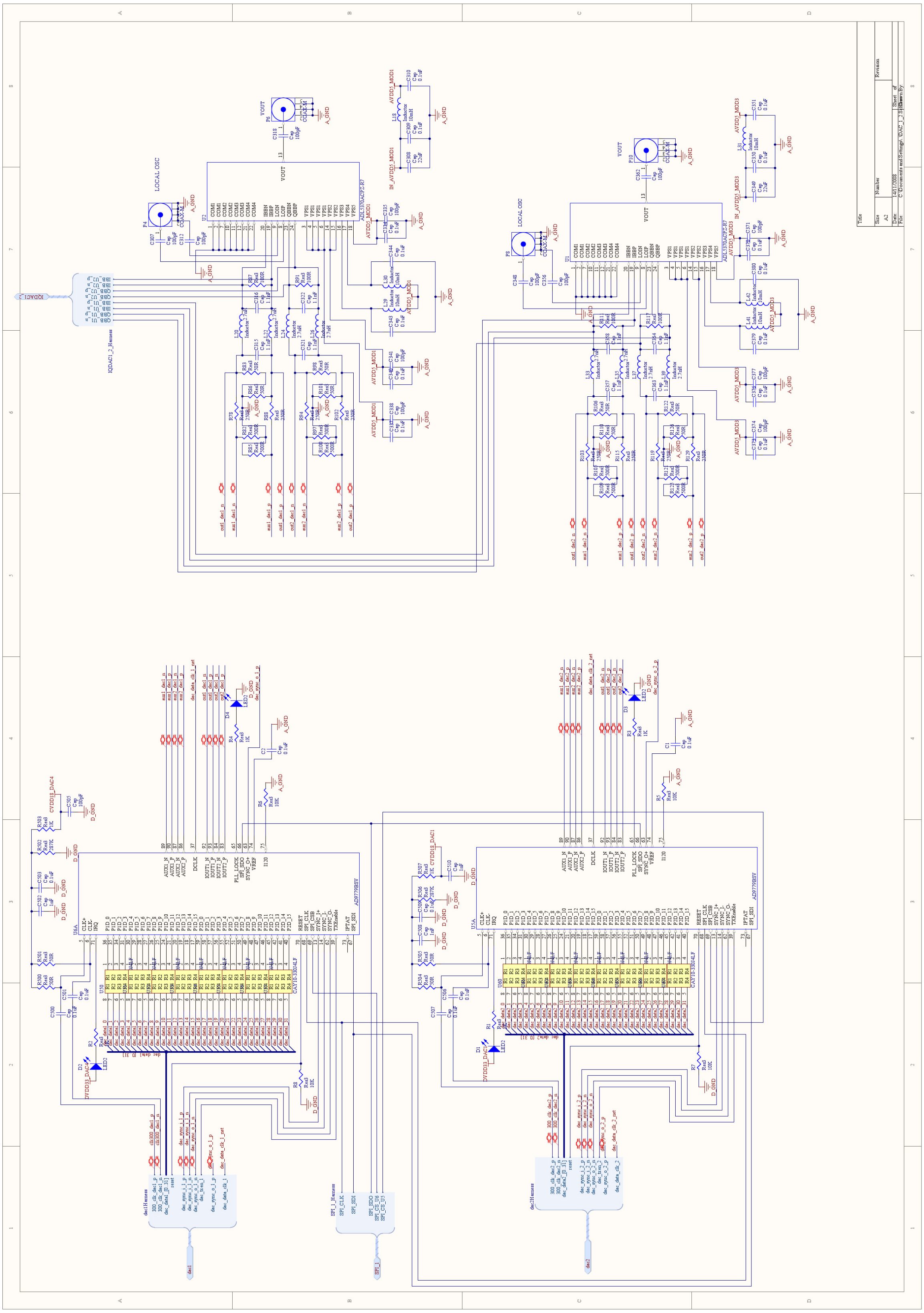

Figure B.2: Schematic page 2 - DAC 1/2 and IQ modulator 1/2

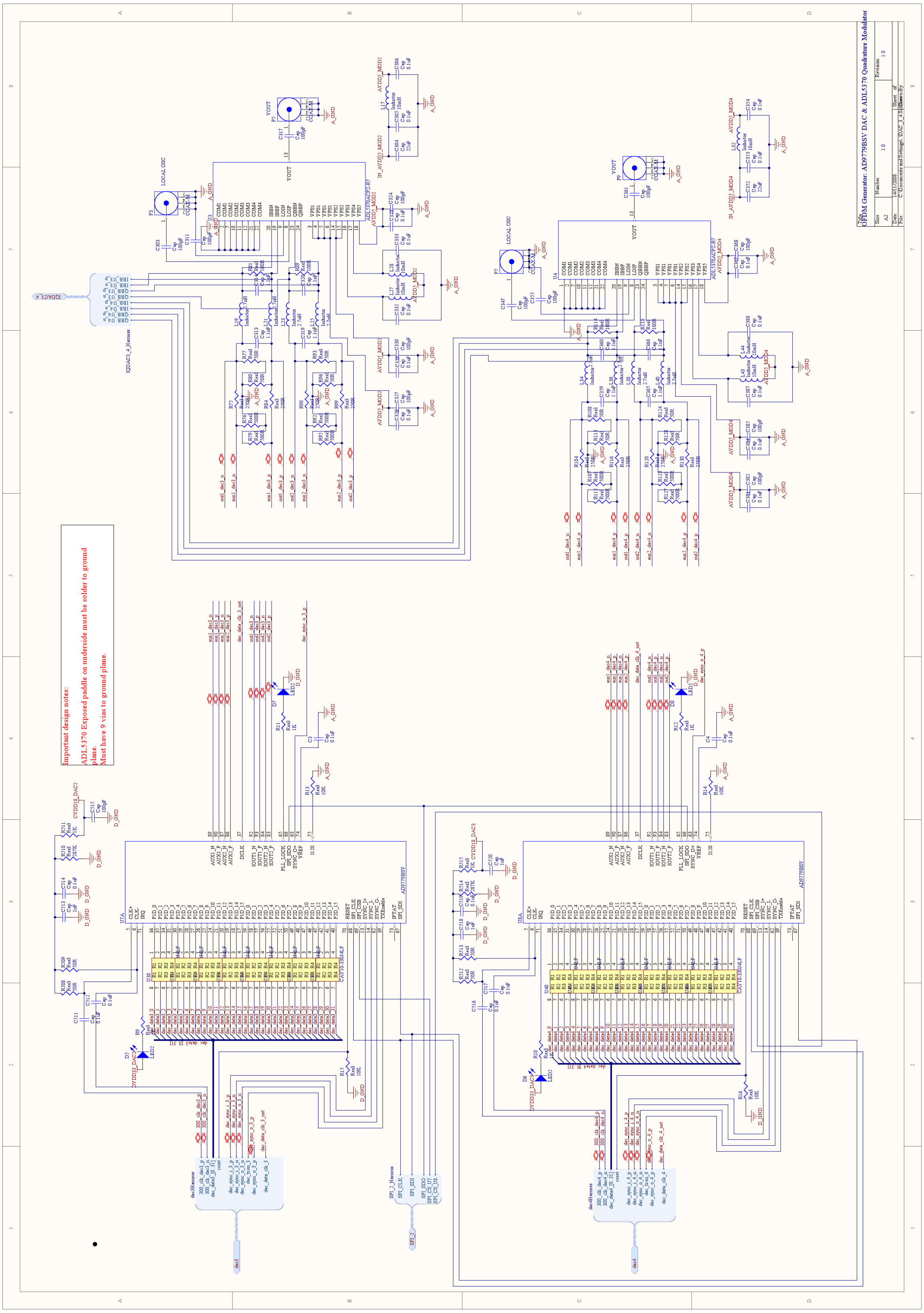

Figure B.3: Schematic page 3 - DAC 3/4 and IQ modulator 3/4

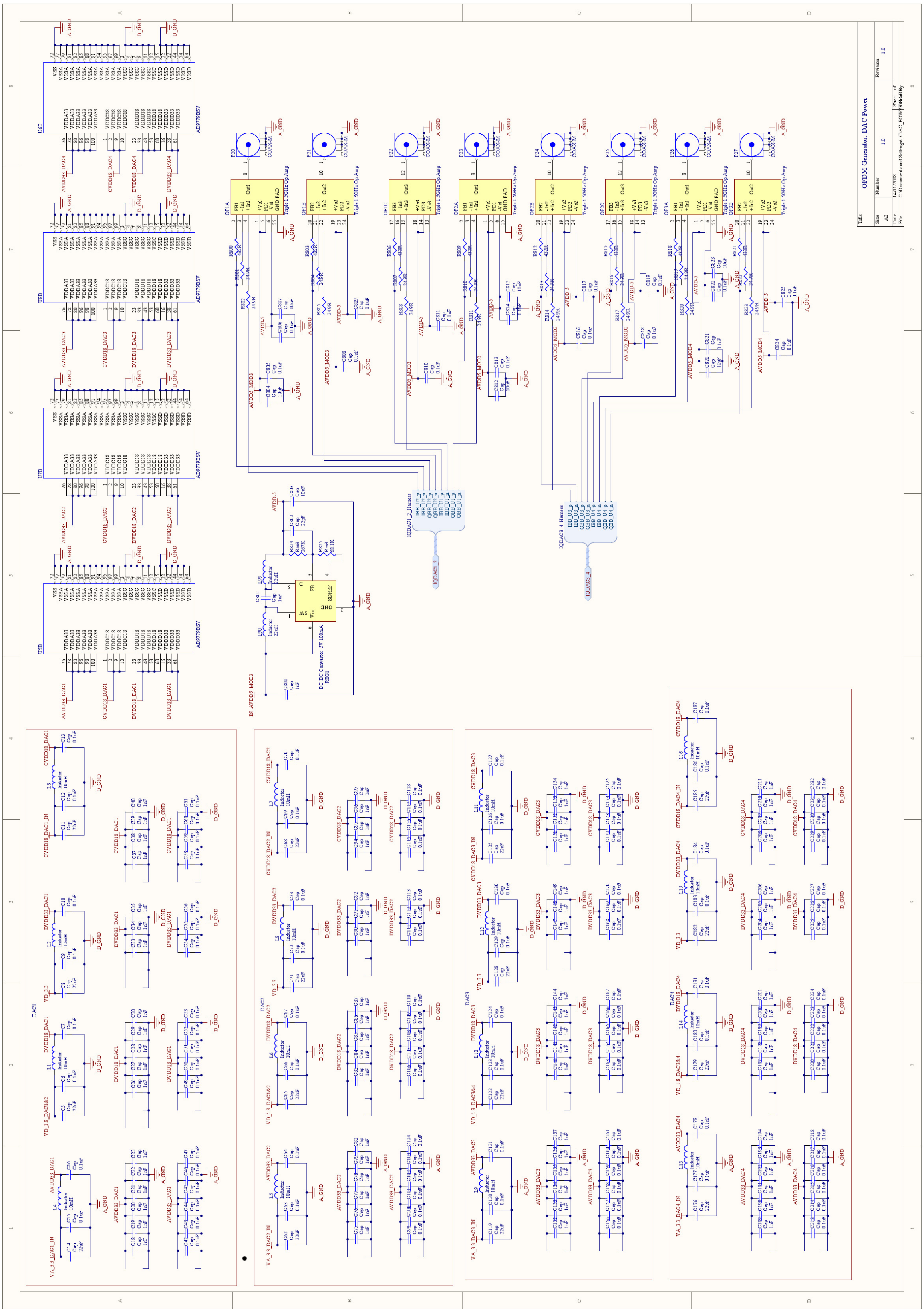

Figure B.4: Schematic page 4 - DAC power supplies and decoupling

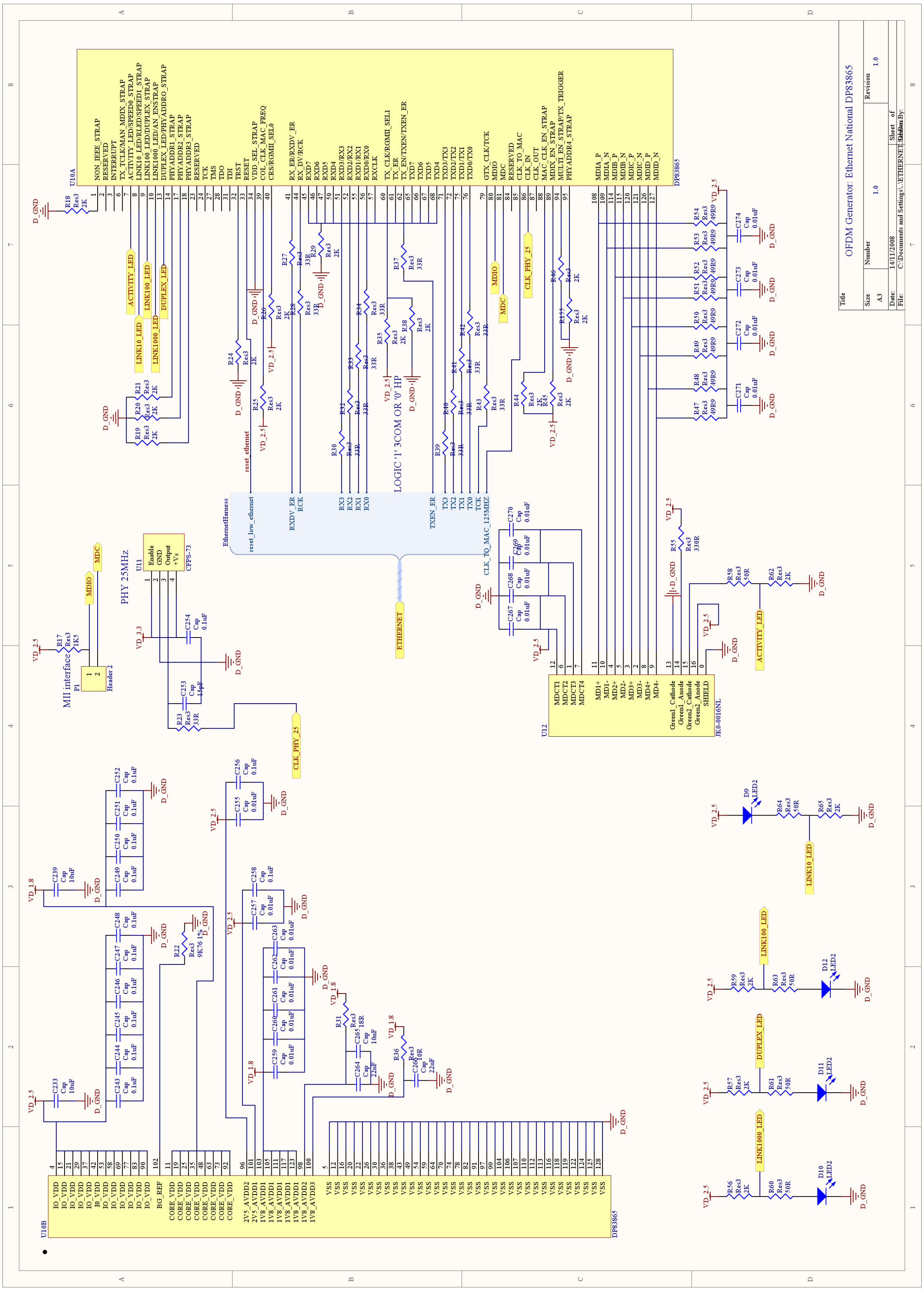

Figure B.5: Schematic page 5 - Ethernet PHYceiver

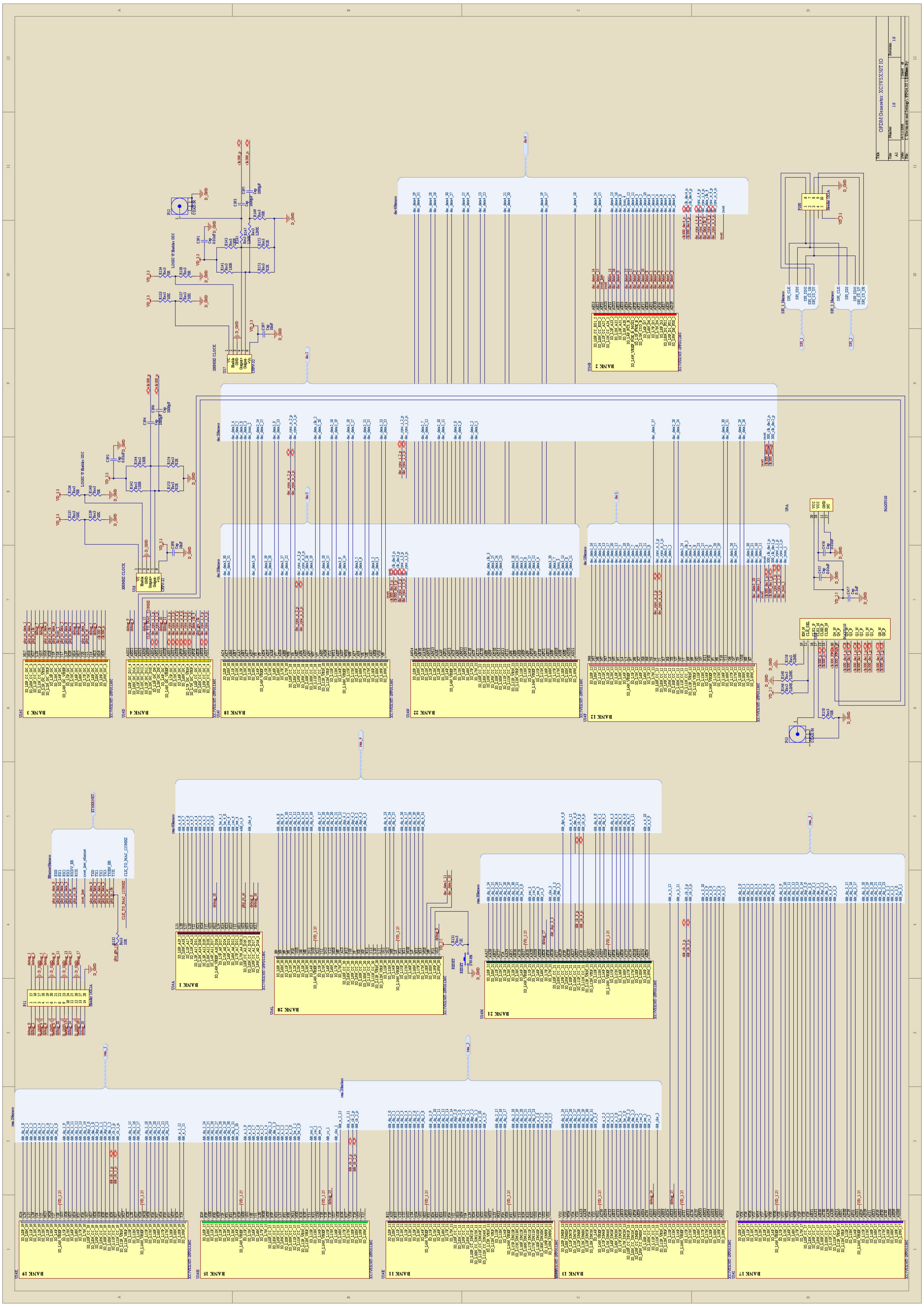

Figure B.6: Schematic page 6 - FPGA I/O connectivity

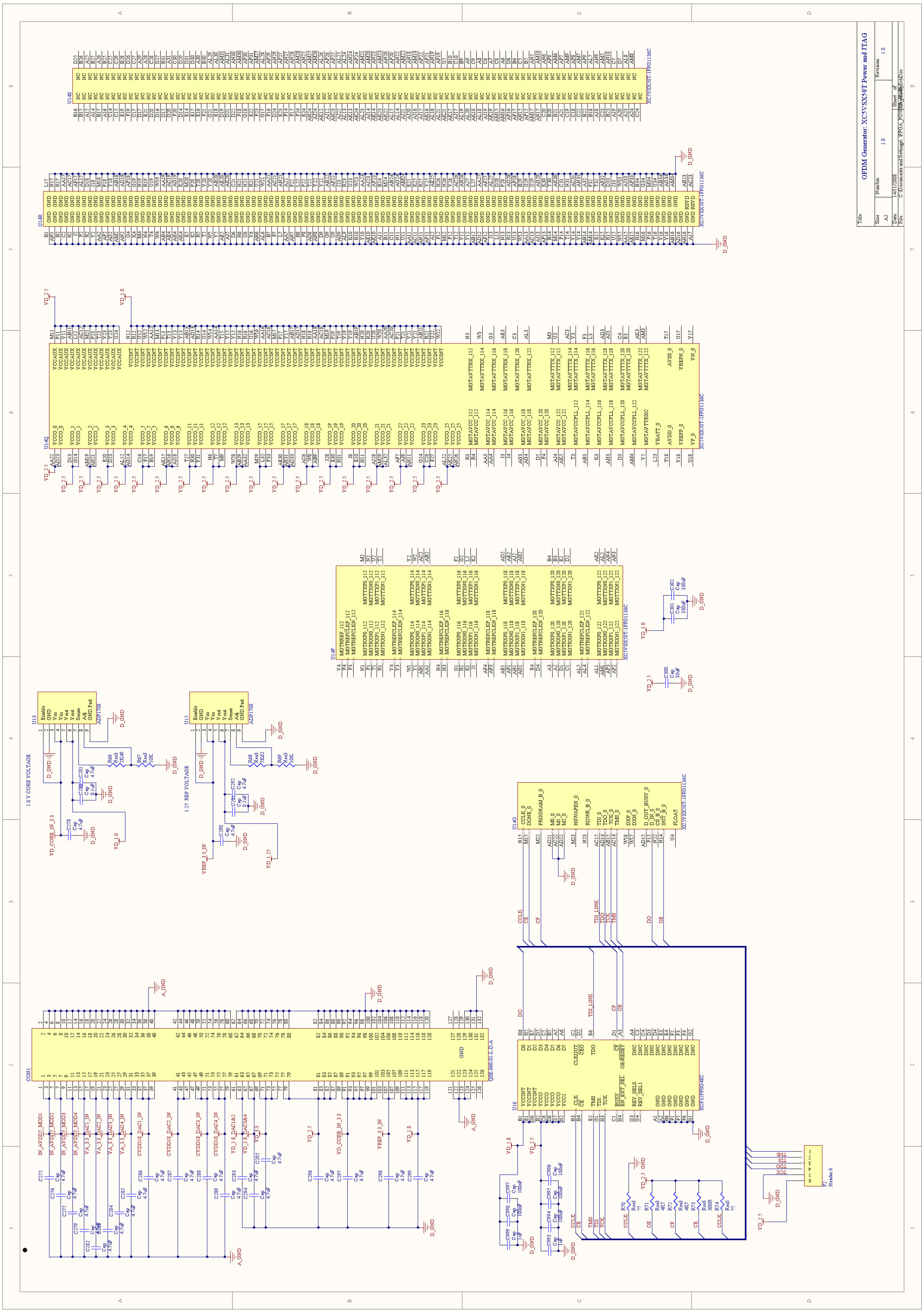

Figure B.7: Schematic page 7 - FPGA flash and JTAG interface

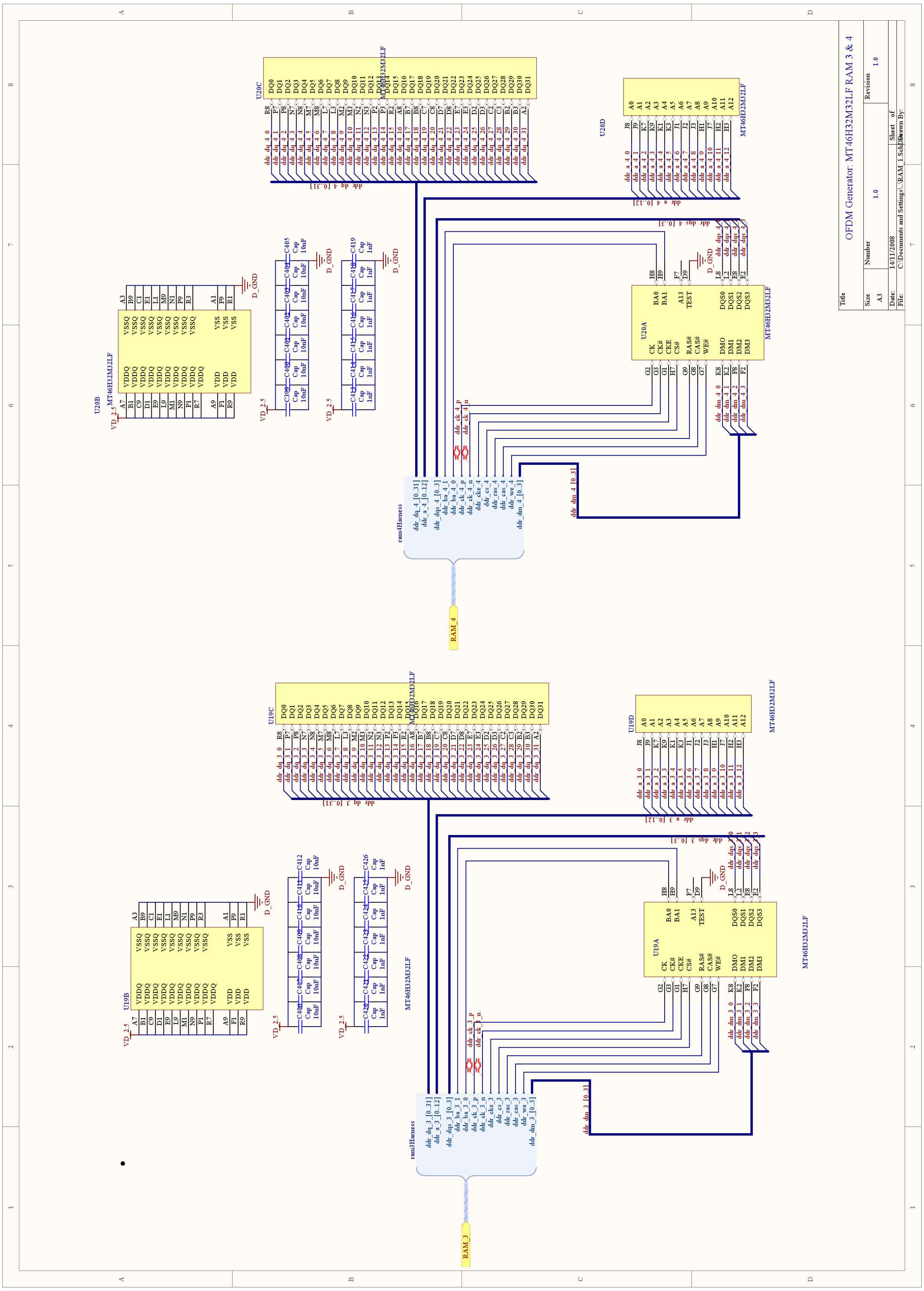

Figure B.8: Schematic page 8 - DDR RAM 3/4

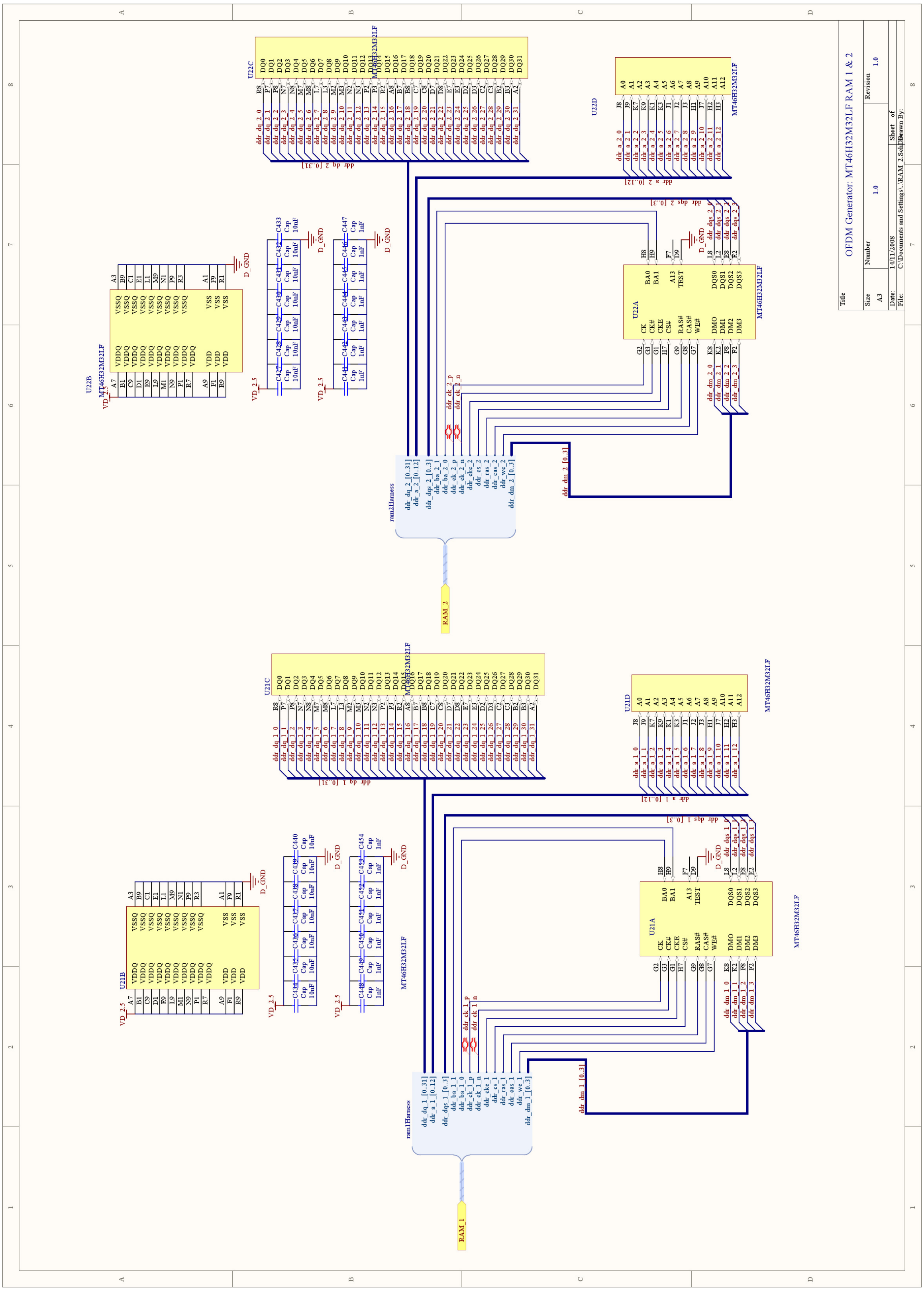

Figure B.9: Schematic page 9 - DDR Ram 1/2

Appendix C

PCB Gerber Files

This appendix contains the source of the Printed Circuit Board (PCB), designed as part of the Field Programmable Gate Array (FPGA) wireless signal generator platform detailed in Chapter 2. The files are graphical representations of X/Y co-ordinate data (defined as Gerber or 247X) that is sent out to a manufacturer to produce the physical PCB. This PCB has 24 layers and contains over 2500 components.

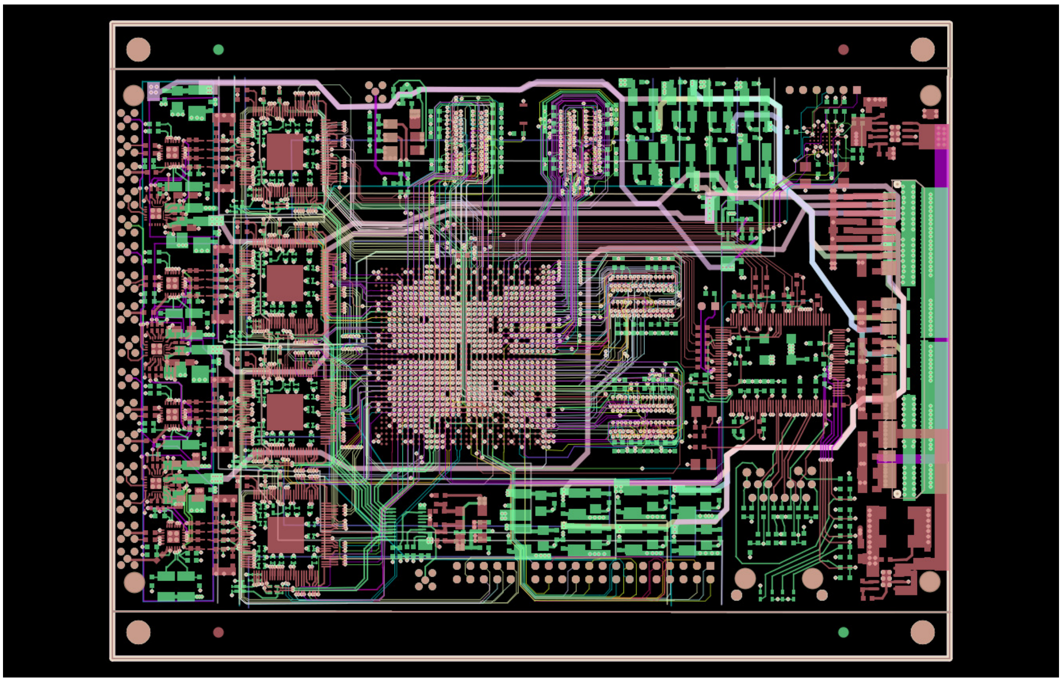

Figure C.1: All 24-PCB layers together

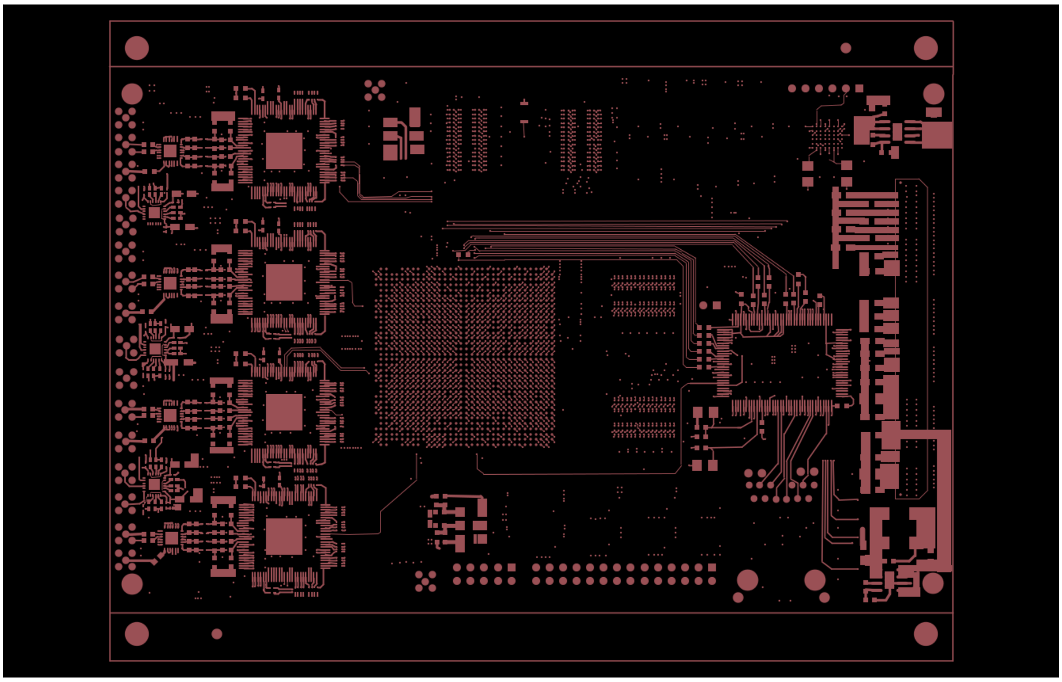

Figure C.2: Top (signal) layer

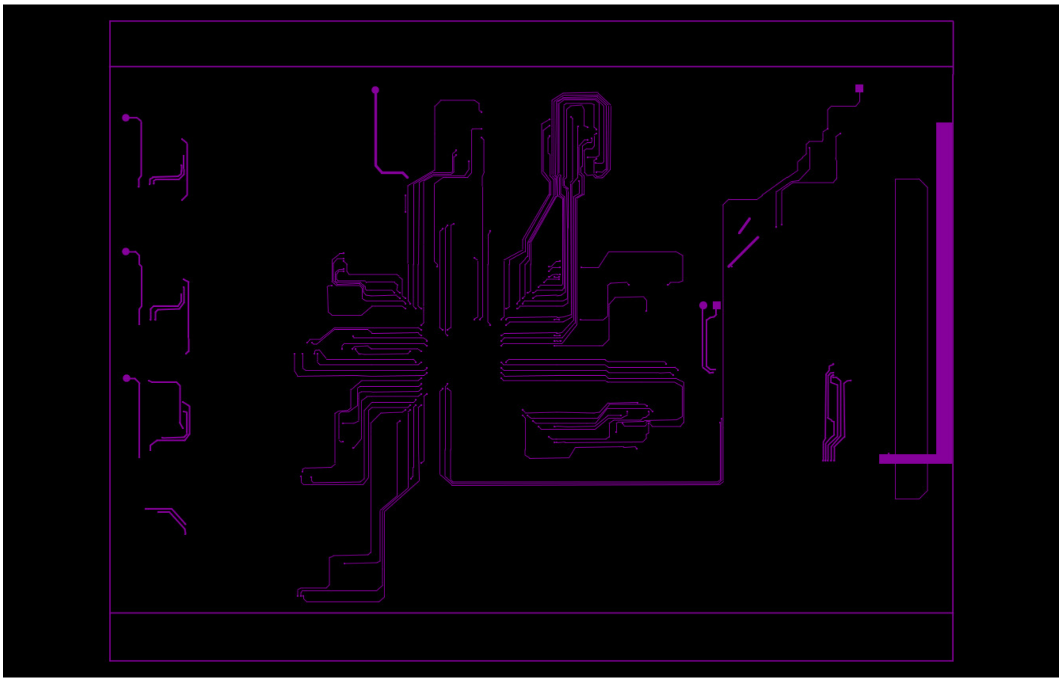

Figure C.3: Signal layer 1

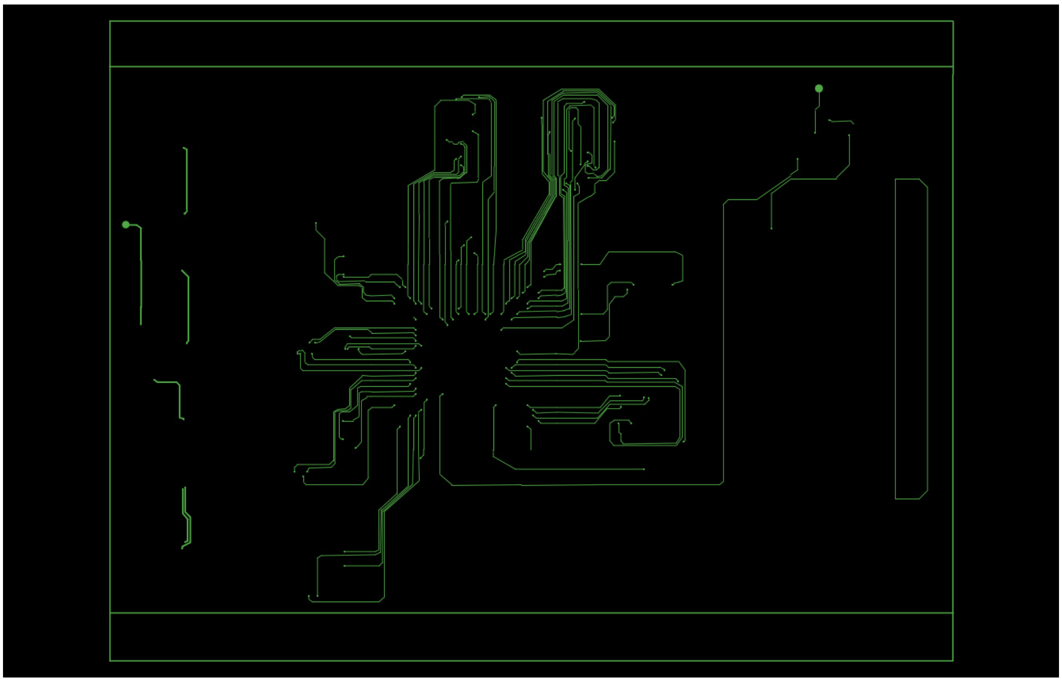

Figure C.4: Signal layer 2

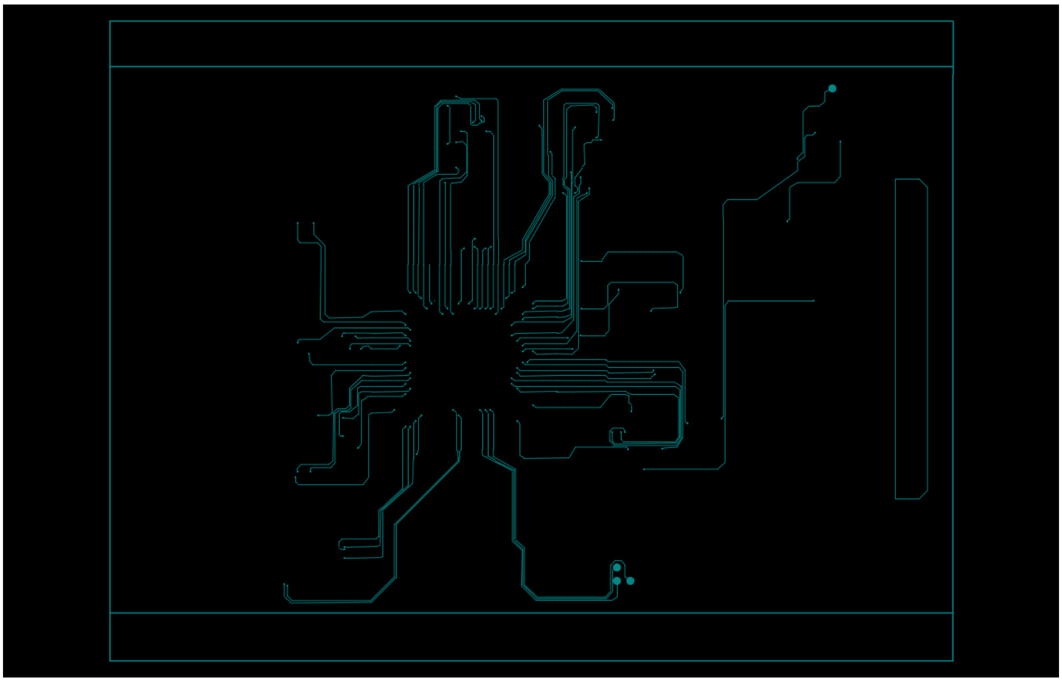

Figure C.5: Signal layer 3

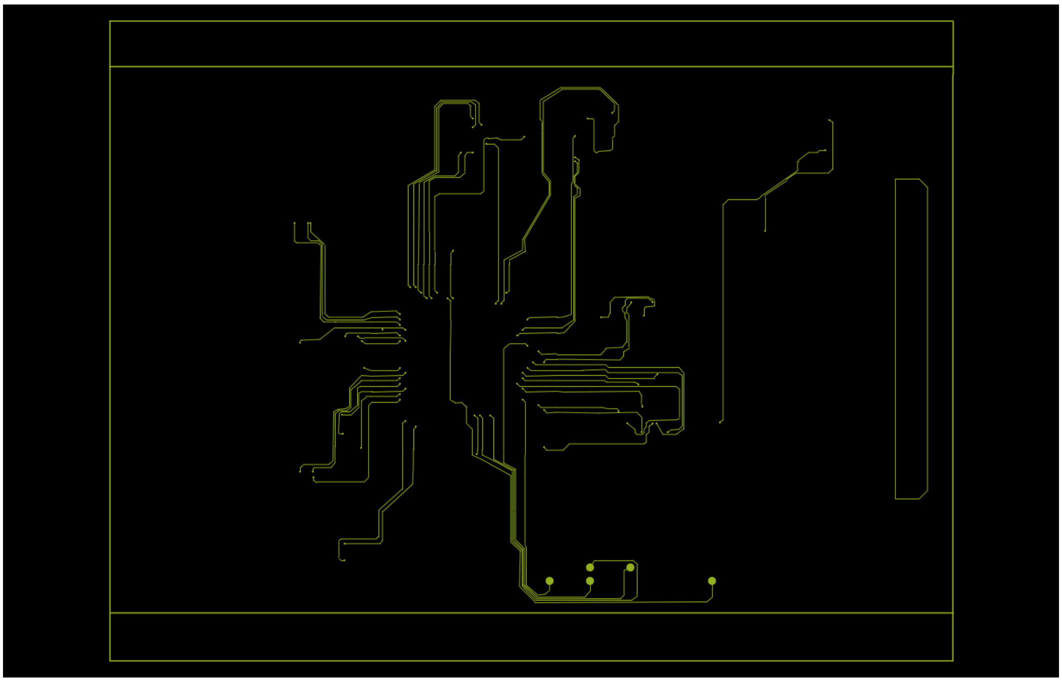

Figure C.6: Signal layer 4

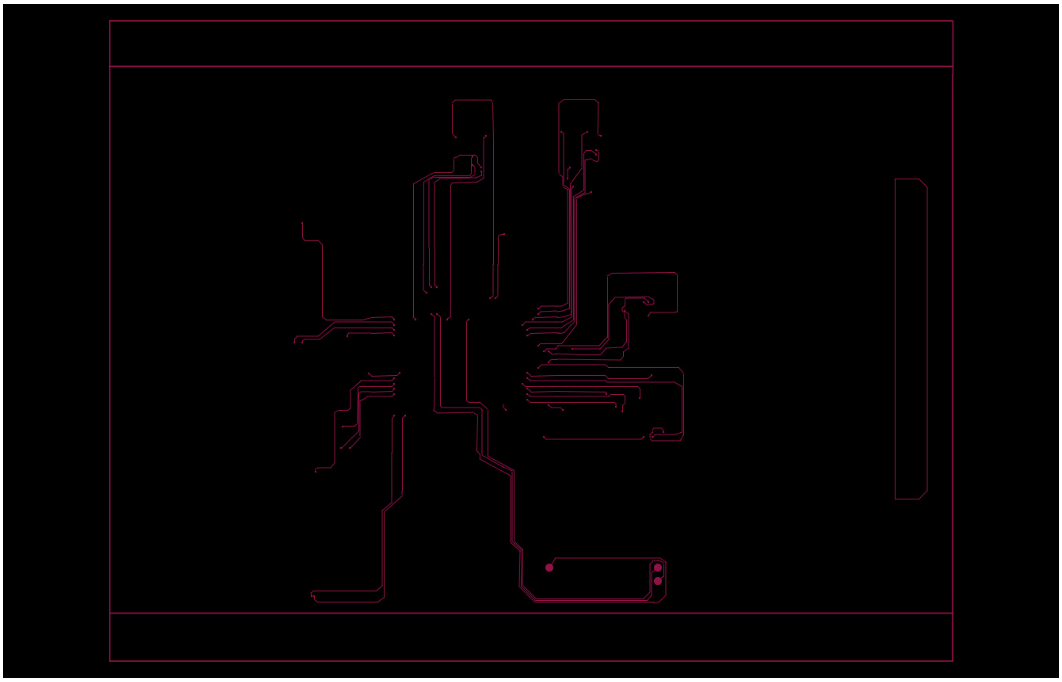

Figure C.7: Signal layer 5

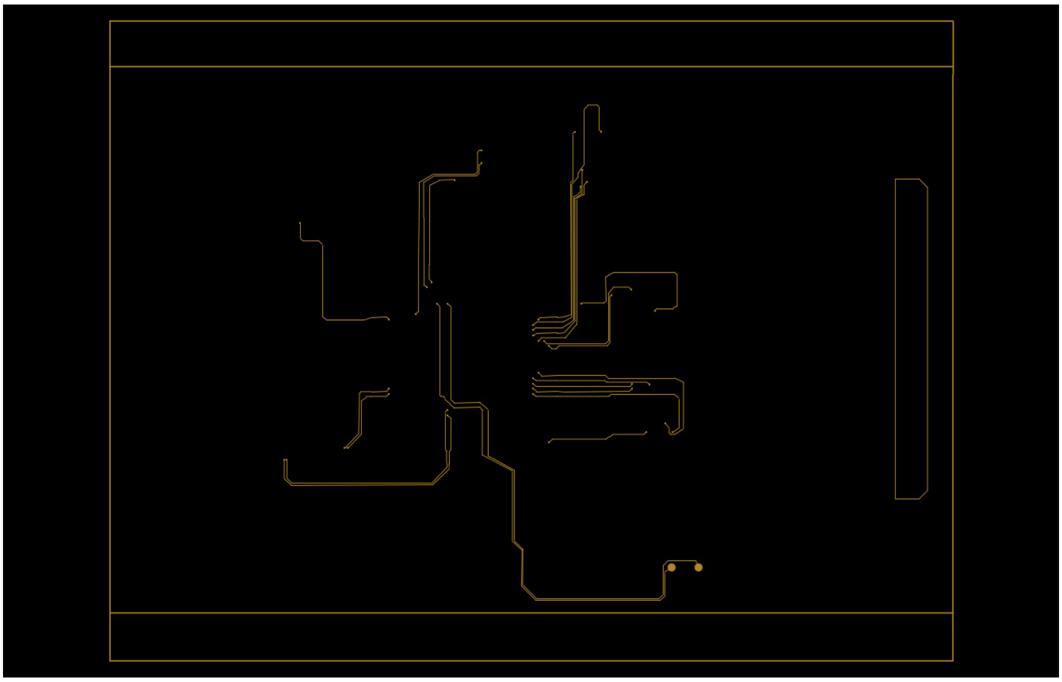

Figure C.8: Signal layer 6

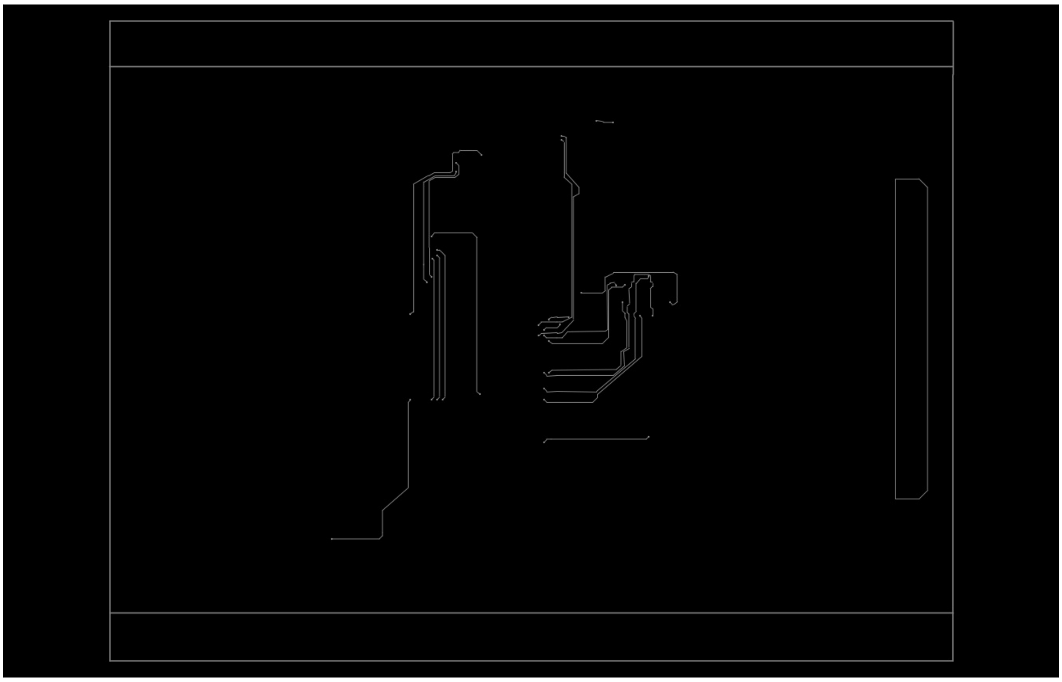

Figure C.9: Signal layer 7

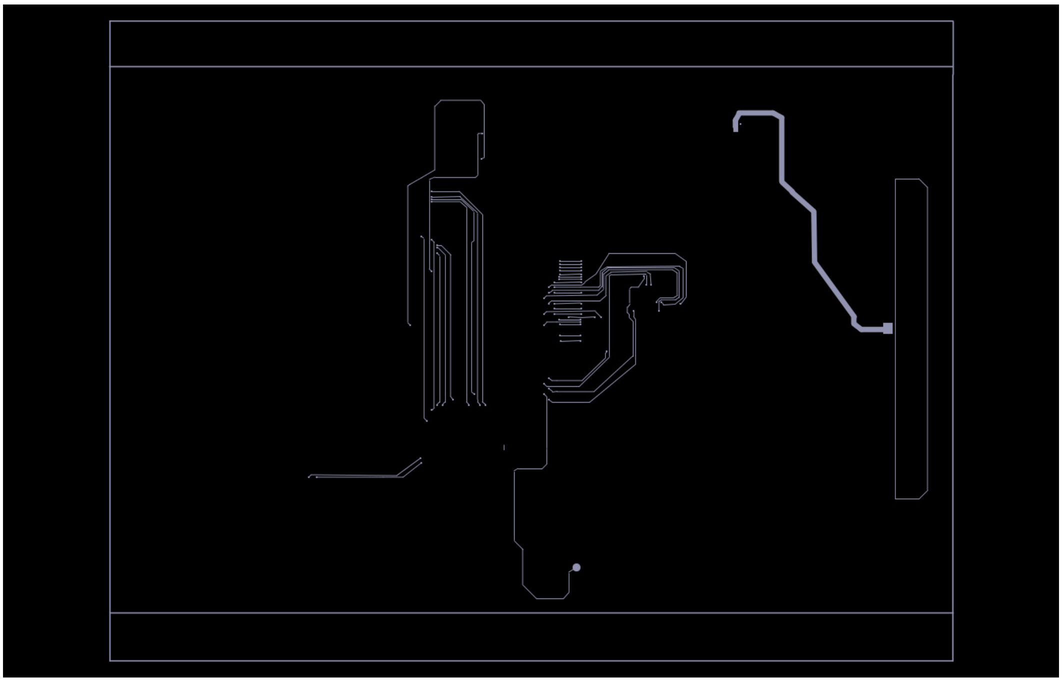

Figure C.10: Signal layer 8

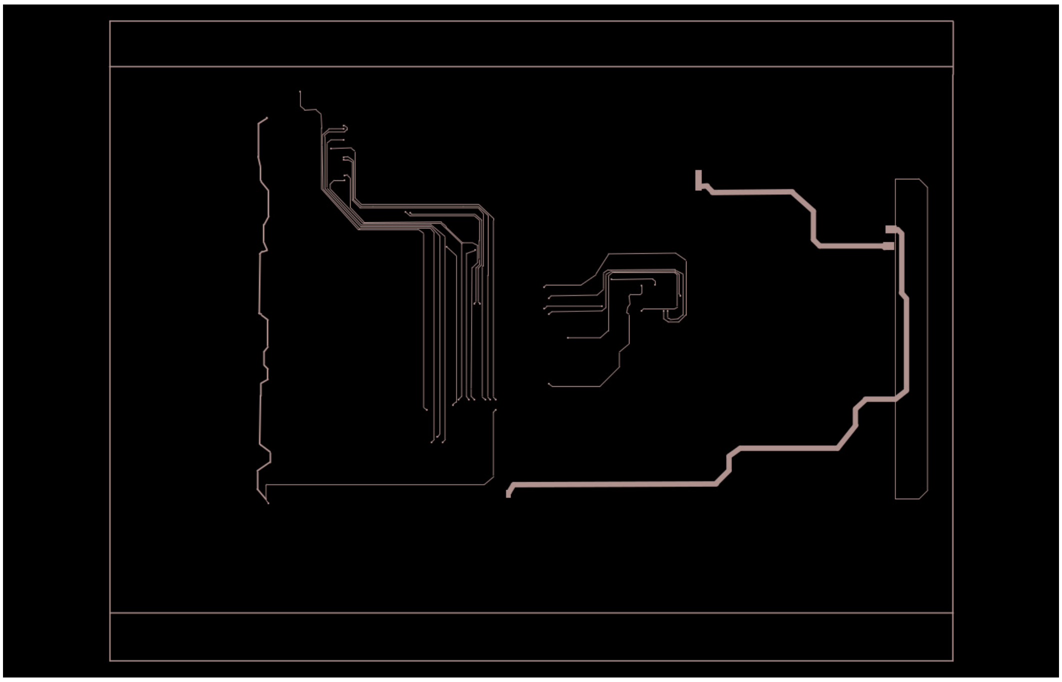

Figure C.11: Signal layer 9

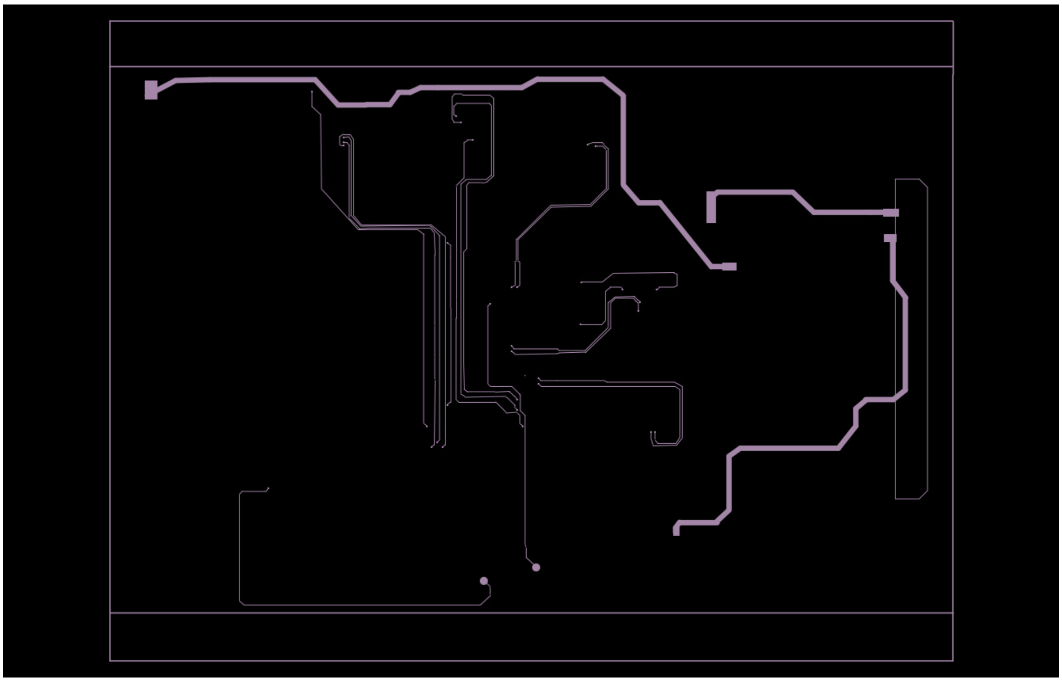

Figure C.12: Signal layer 10

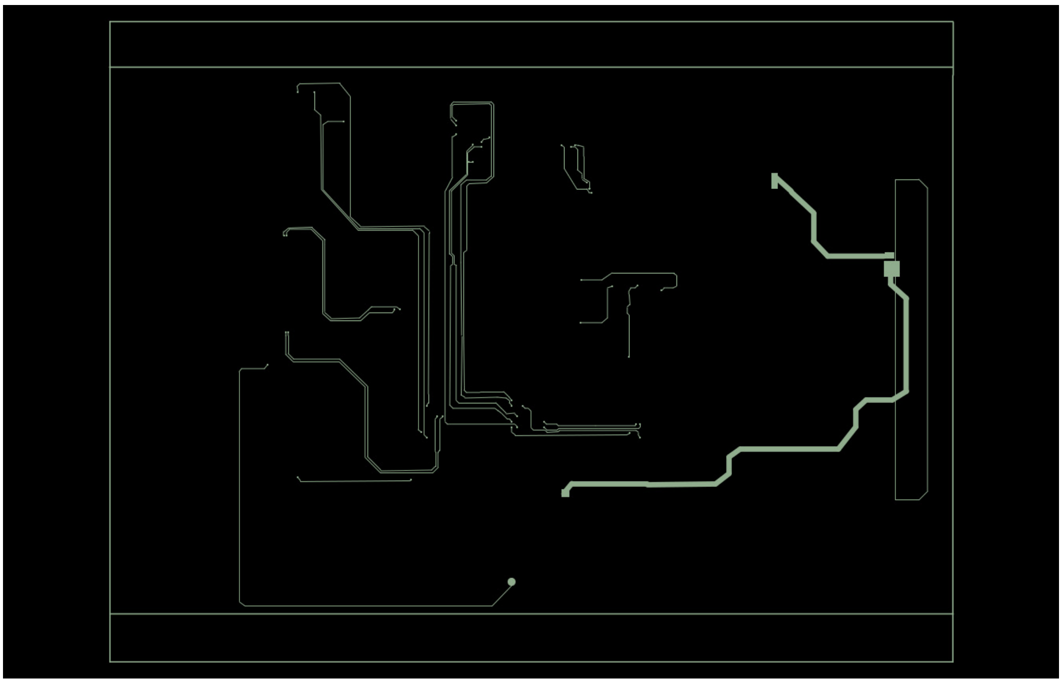

Figure C.13: Signal layer 11

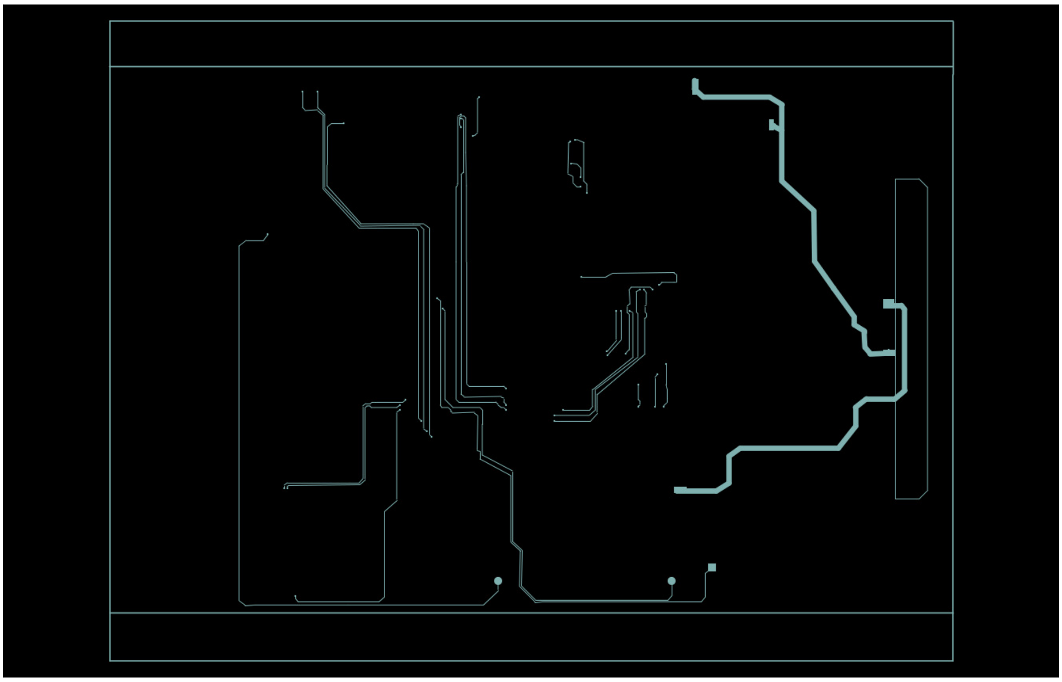

Figure C.14: Signal layer 12

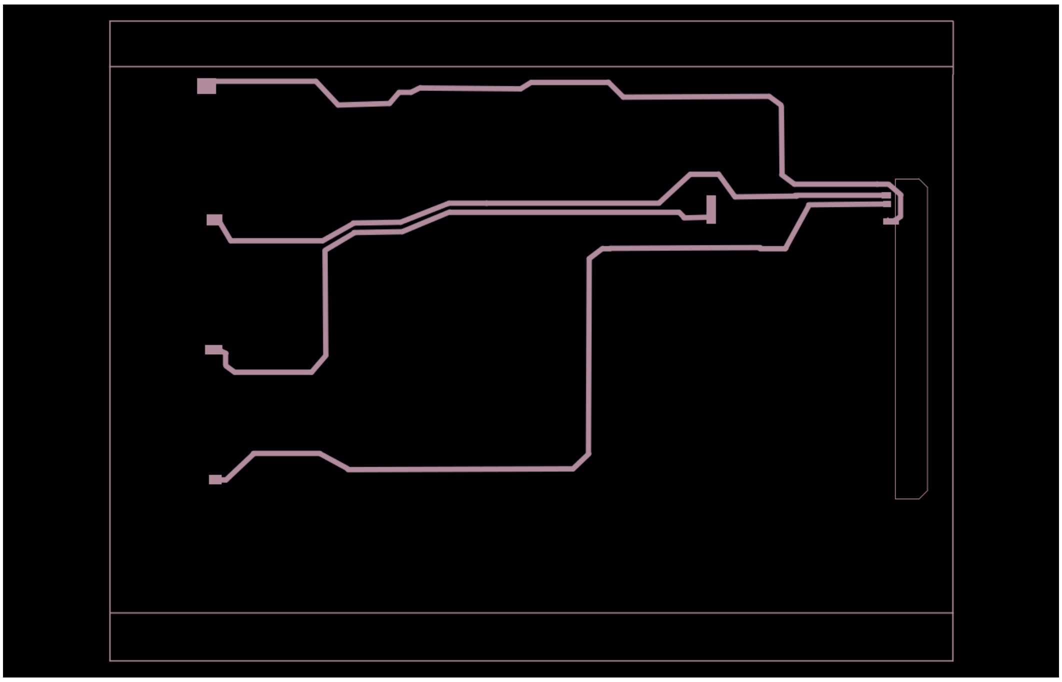

Figure C.15: Signal layer 13

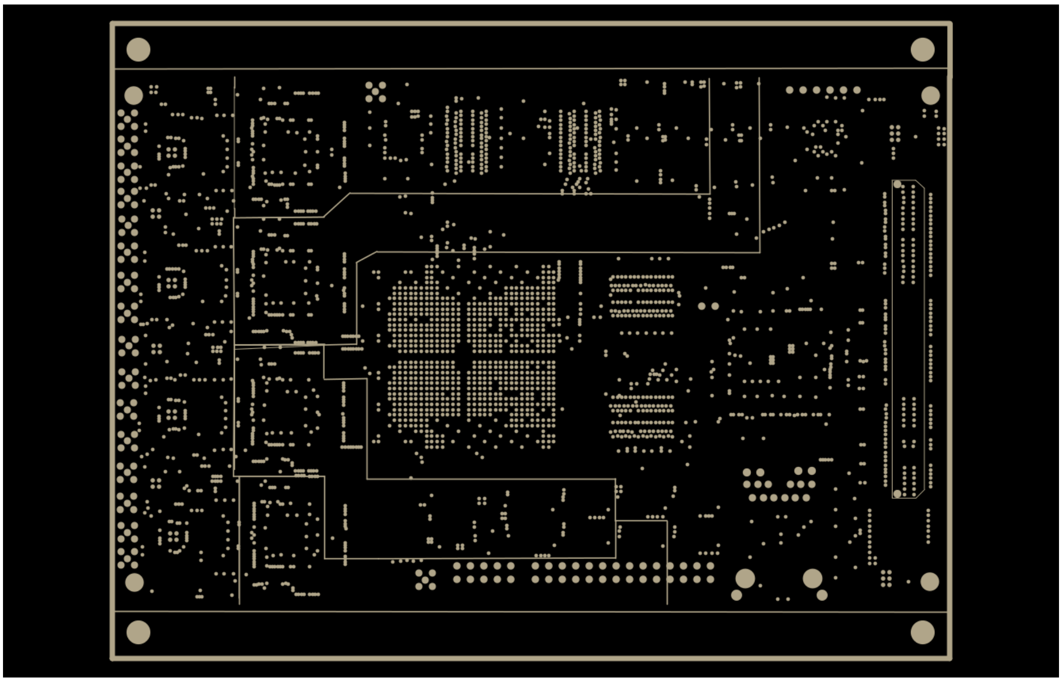

Figure C.16: Power layer 1

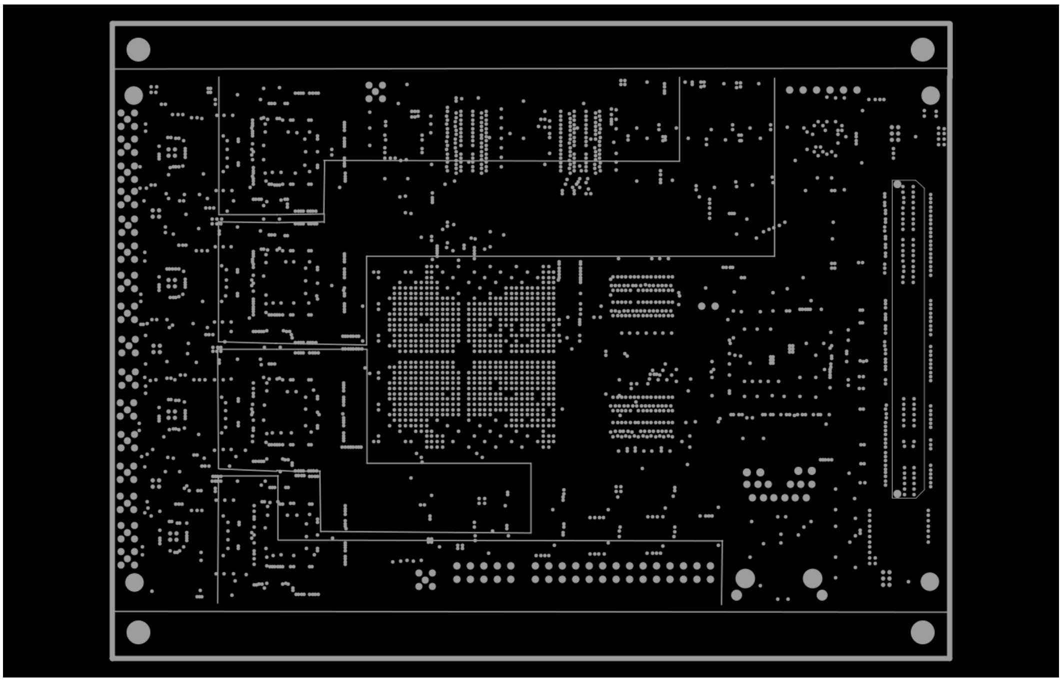

Figure C.17: Power layer 2

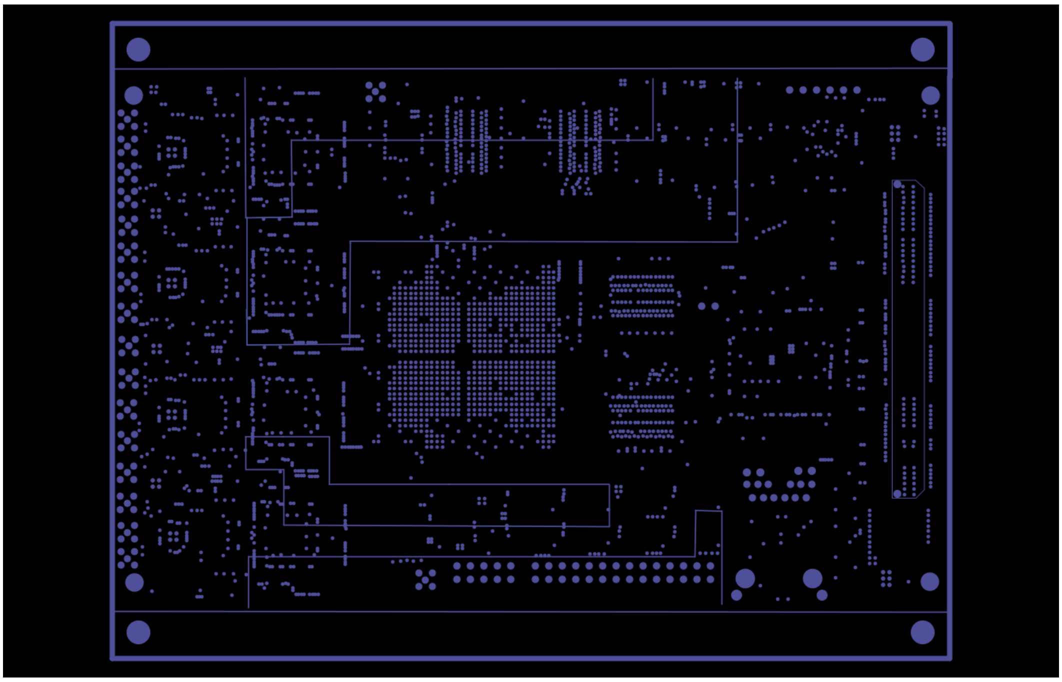

Figure C.18: Power layer 3

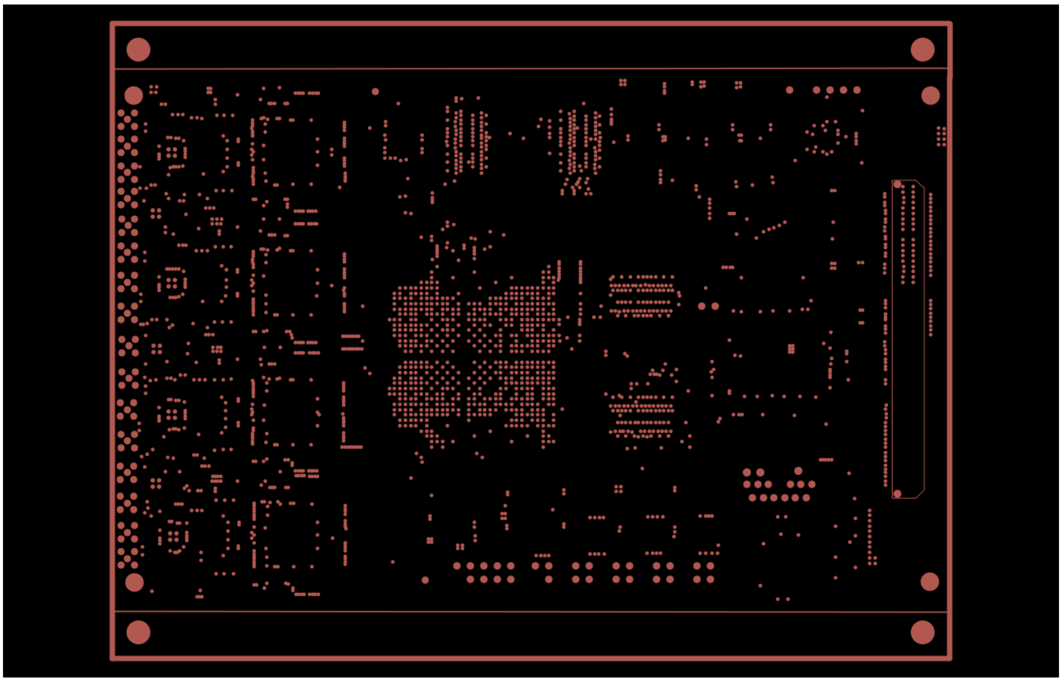

Figure C.19: Power layer 4

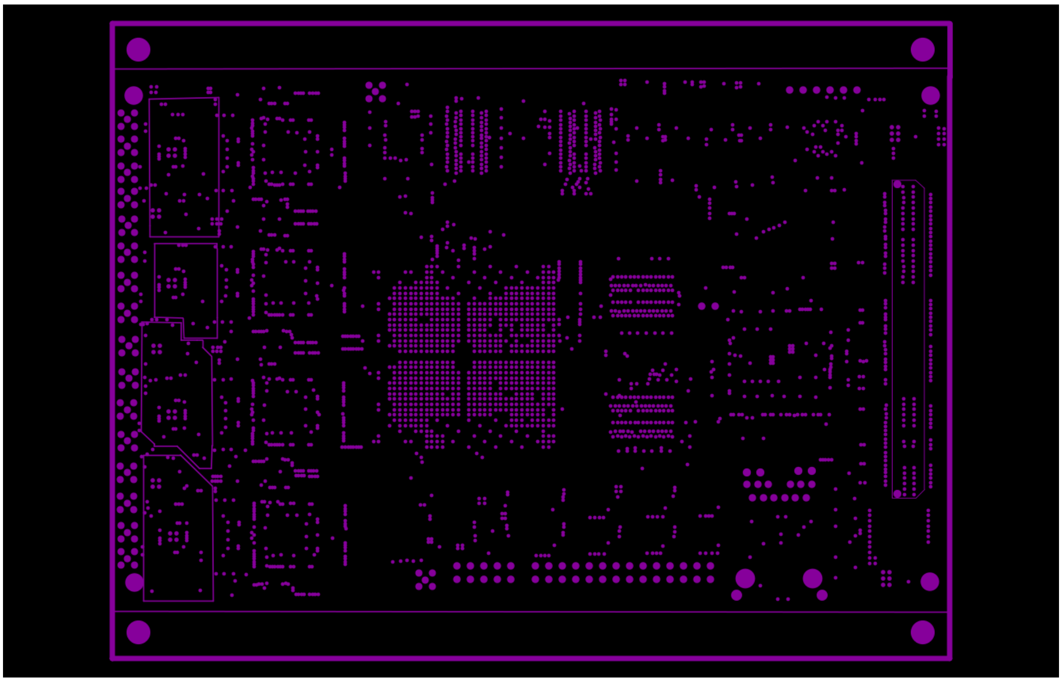

Figure C.20: Power layer 5

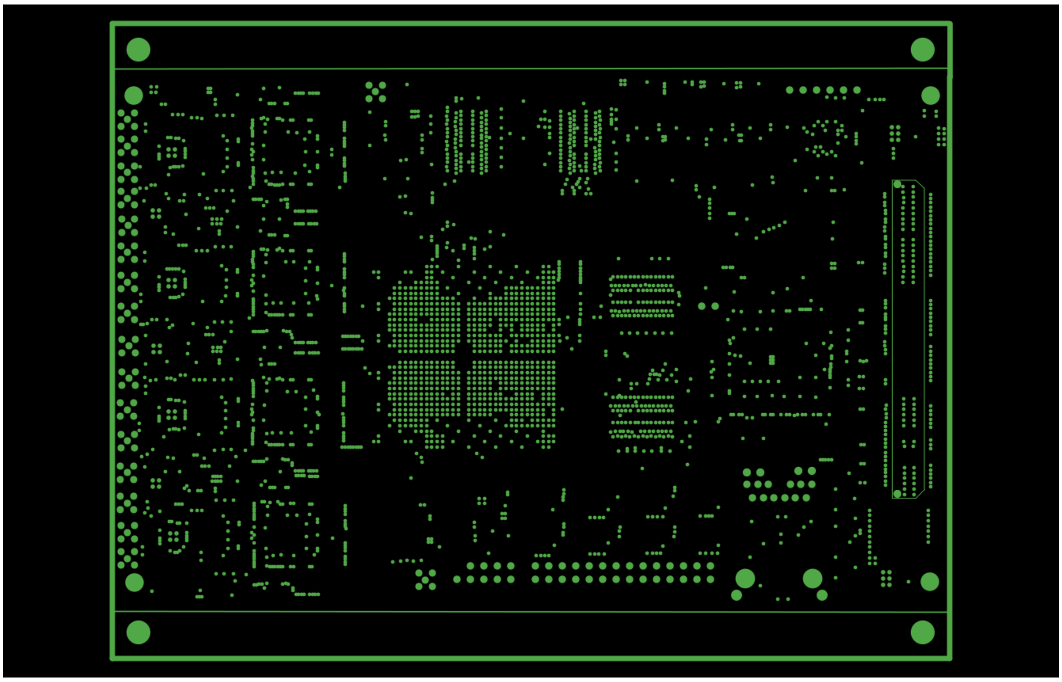

Figure C.21: Power layer 6

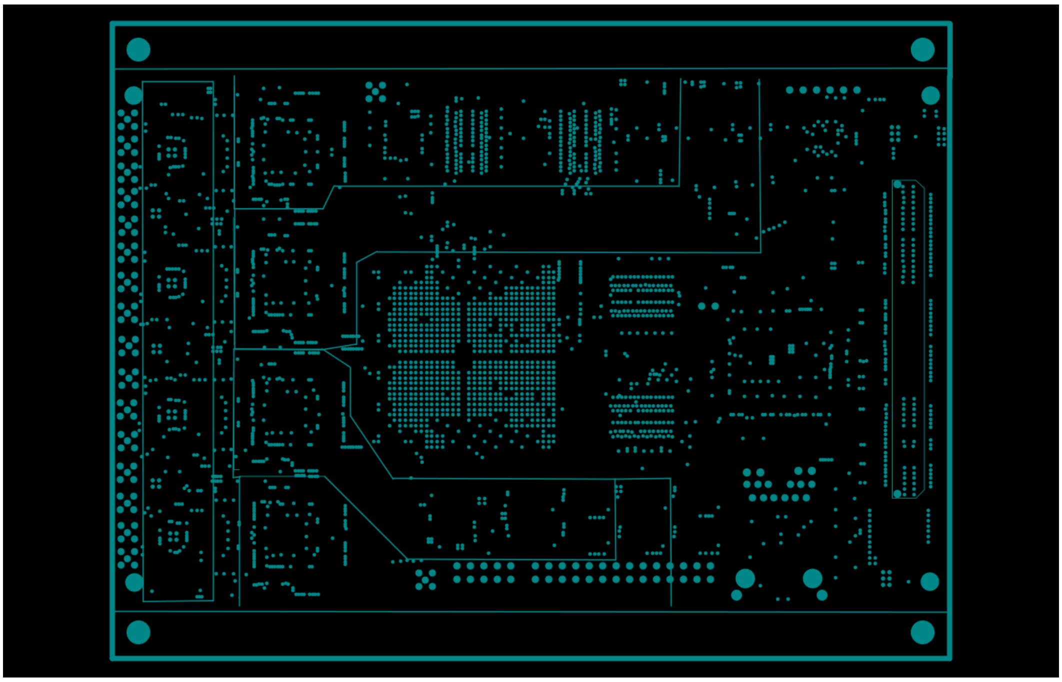

Figure C.22: Power layer 7

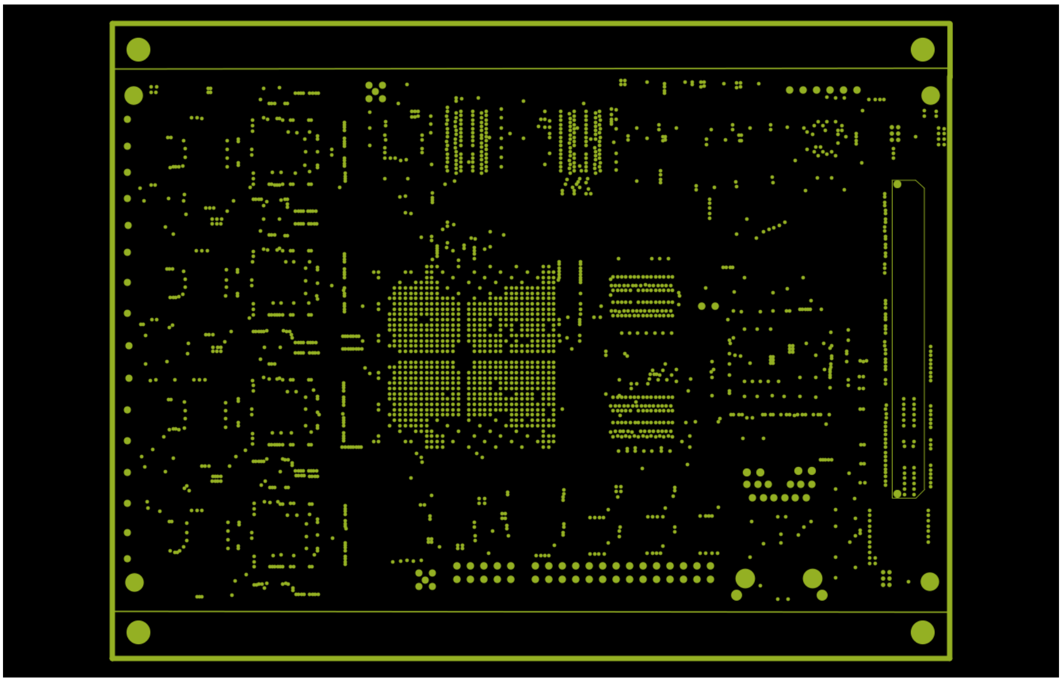

Figure C.23: Power layer 8

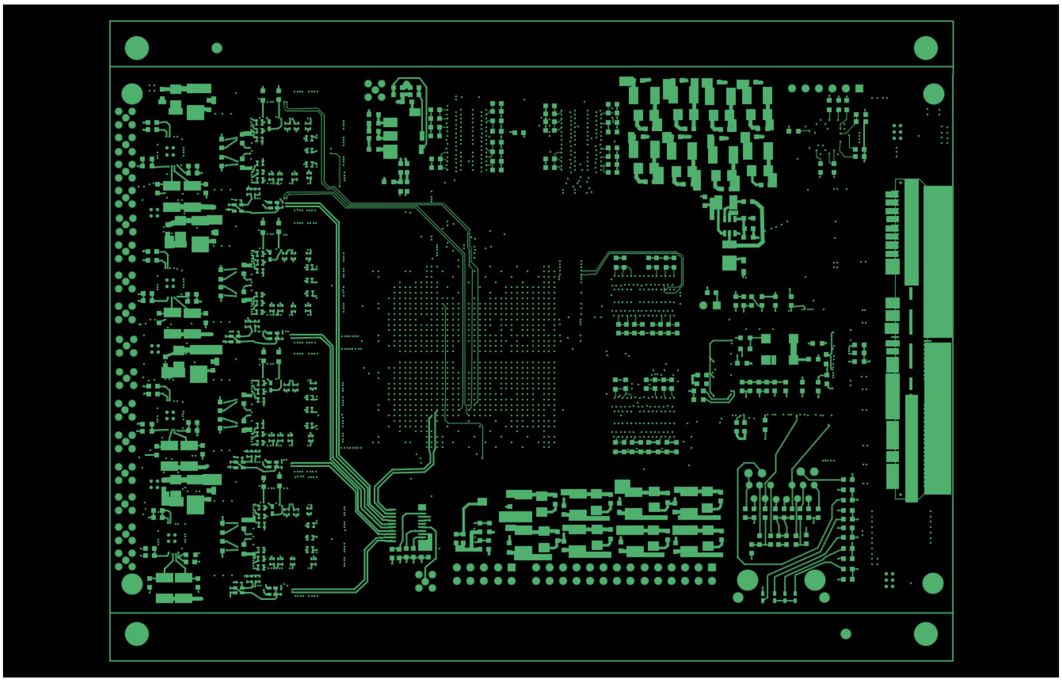

Figure C.24: Bottom (signal) layer

Appendix D

SEFDM Generator GUI Datasheet

This document details a Graphical User Interface (GUI) that has been developed to provide control and supply data to a custom Spectrally Efficient Frequency Division Multiplexing (SEFDM) generation platform. The GUI is written using C++ within Microsoft Visual Studio on a core i7 laptop. Conceptually, control and source data from the GUI are transmitted over Ethernet to a custom designed Field Programmable Gate Array (FPGA)-based platform, utilising a custom design stack to enable simple communications. This platform has logic pre-loaded and can generate signals digitally. These signals are converted to analog using on-board Digital to Analogue Converters (DACs), which can be viewed and analysed using standard test equipment. A detailed description of the operations performed by the GUI code, together with the physical Ethernet interface and the reconfiguration operations performed by the FPGA platform, is presented in Figure D.1.

In detail, operations from the GUI (described in more detail in the following section(s)) are encapsulated in standard networking protocols, before passing data to the operating system for transmission using a Network Interface Card (NIC). The top left of Figure D.1 is the reception of such signals within the FPGA system, which are subject to standard Ethernet processes from the PHYCeiver and Media Access Controller (MAC). Assuming the output of these processes is correct (i.e the data payload has no errors), the data is made available to the FPGA internal logic. Here, a

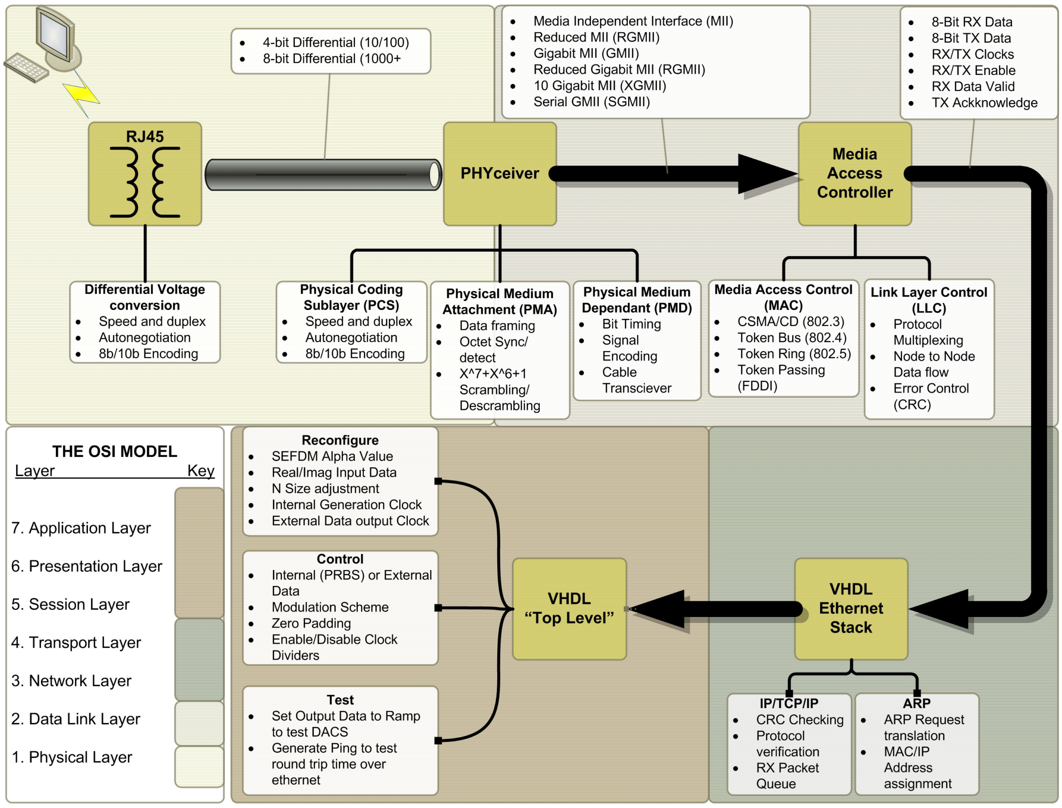

Figure D.1: Extended Ethernet diagram introducing the Stack and Top Level interfaces

custom stack (which is detailed in Chapter 4) determines if the payload is an Address Resolution Protocol (ARP) or a command for reconfiguration purposes. If the payload is the former, the stack replies with the appropriate pre-formatted message. If the payload is the latter, it is passed to the area of the FPGA responsible for SEFDM generation (defined as *top level*).

The reconfiguration of the SEFDM transmitter is three-fold. The first, **Reconfigure**, allows internal variables that determine the characteristics of the resulting SEFDM signal to be modified. Parameters such as N (the number of sub-carriers), the spacing between such sub-carriers (α, which is detailed in Chapter 5) and internal and external clock frequencies, can be modified. The second, **Control**, are binary operations by which the processes modified are fixed but have controllable elements. Options include the modulation scheme applied to source data internally and enabling and disabling of zero padding. The third, **Test**, enables the output of standard and predefined signals, which can be used for test or debugging purposes. Such signals include sine, ramp and narrow pulse (impulse) outputs, as detailed in Chapter 3.

In the following, each function of the GUI is described in detail. Upon starting the application the GUI is displayed with values defaulted to those as in Figure D.2. At the same time, a connection is established to the Ethernet port that the application is running on. Such a connection is pre-configured within the application to match parameters expected by the FPGA platform. Each of the GUI features is described

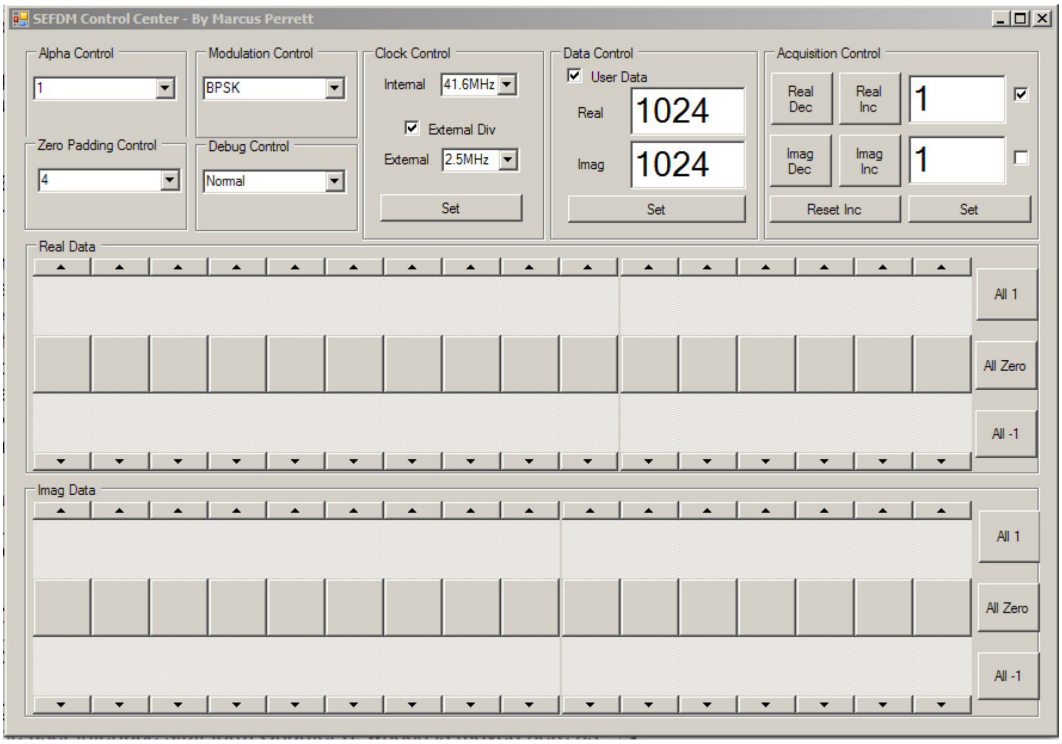

Figure D.2: Control GUI default startup screen

in more detail in the following sections:

Alpha

The value of Alpha corresponds to a pre-determined ratio of two values, b and c. When they are evaluated as $\frac{b}{c}$, the resulting value, known as Alpha (α), determines the relationship between sub-carriers. For example, $\alpha = 1$ would be Orthogonal Frequency Division Multiplexing (OFDM) and $\alpha < 1$ would be some value of SEFDM.

The GUI is illustrated in Figure D.3. The available values of *alpha* are determined

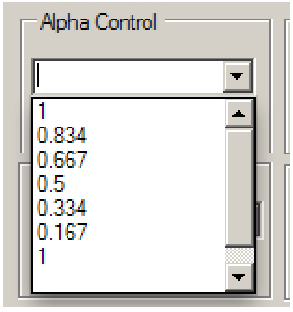

Figure D.3: Alpha control area showing available sub-carrier compression fractions

by the logic that exists within the FPGA, as described in Chapter 5. Due to resource constraints and implementation limits (which are also described in Chapter 5), only values where $c = 3$ or $c = 6$ are available. Therefore, 0.834 as a ratio of $b = 5$ and $c = 6$ is available, but 0.75 as a ratio of $b = 3$ and $c = 4$ is not available. Furthermore, some values (such as 0.167) are practically unrealistic but technically possible and are therefore included.

When *alpha* is changed, the internal interconnects that determine the SEFDM generation internal to the FPGA are modified appropriately. Furthermore, values required for producing a frequency shift at the output are re-generated. As a single block is used (within the FPGA) to generate such values, a period on inactivity will ensue while these values are created. The actual delay time depends on the value of c and the internal clock frequency the logic is running at.

Zero Padding

The zero padding determines the number of source data inputs that are unused. Those remaining in use translate directly to the sub-carriers that will form the SEFDM output. For example, if $N = 16$ and Zero Padding is set to 4, the resulting OFDM or SEFDM symbol would consist of 8 sub-carriers (16 - 4 - 4), distributed evenly from a central point determined by $N/2$. The available zero padding values are shown in Figure D.4 As zero padding is applied to each side of the Inverse Fast Fourier

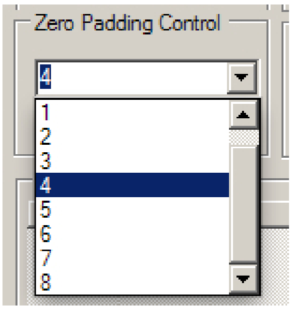

Figure D.4: Zero padding area showing available zero padding values. Padding is applied to both ends of the spectrum

Transform (IFFT) transform(s), a maximum of 8 can be set. The is because in current version, N is fixed at 16.

Data Modulation

Internal to the FPGA, modulation can be set and performed on the input data. Each binary-represented-decimal number corresponds to a specific scheme, thus 0=Binary Phase Shift Keying (BPSK), 1=Quadrature Phase Shift Keying (QPSK), 2=8-Quadrature Amplitude Modulation (8-QAM), 3=16-Quadrature Amplitude Modulation (16-QAM) etc. Available modulation schemes are displayed in Figure D.5.

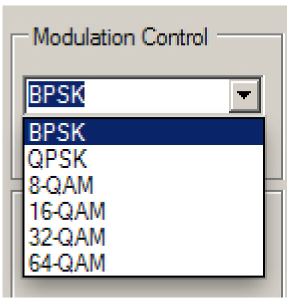

Figure D.5: Modulation control area showing available modulation schemes available

Debug

As mentioned above, pre-determined signals can be generated by the FPGA for debug or characterisation purposes. The available signals are illustrated in Figure D.6. The use of these signals are used as part of a verification strategy in Chapter 3.

Normal enables SEFDM generation using parameters set by the other parts of the

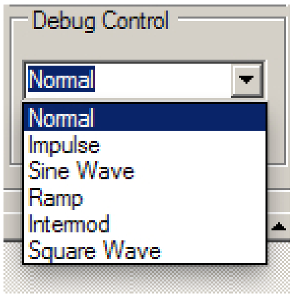

Figure D.6: Debug area showing possible built-in signals that can be generated for characterisation purposes

GUI. *Impulse* generates a single maximum amplitude value once every 2^w, where w is the DAC bit-width value. This can be used for transient or frequency response measurements. *Sine Wave* outputs the contents of memory which is internal to the FPGA. Such memory contains sampled values of a sinusoid and can be used to determine noise measurements. *Ramp* increments the output value to the DAC, from the lowest possible to the highest possible over 2^w and is used to determine output linearity. *Intermod* and *Square Wave* are not currently implemented within the FPGA.

Internal and External Clock

The logic internal to the FPGA can be driven by any of the clock frequencies illustrated in Figure D.7. The FPGA can be re-programmed and may require slower clock speeds due to the size of the design. Generally, this should not be changed from 65.5MHz. On the other hand, the external clock determines the speed at which data to the DAC is output from the FPGA. This is derived using a divider of the DAC source clock. Due to the nature of the divider circuit, the clock can only be divided by numbers which are even. An example of the available external clock frequencies is shown in Figure D.8. The division values implemented are detailed in Table D.1 and are based on a 250MHz source clock. Not all possible values are implemented and instead values which represent a balance between usefulness and complexity are chosen.

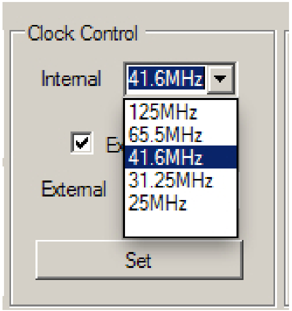

Figure D.7: Internal clock control area showing available clock rates the internal FPGA logic can run

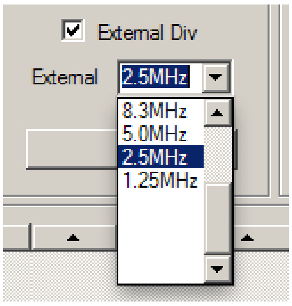

Figure D.8: External clock control area showing available clock rates the FPGA can supply data to the DAC

Table D.1: Possible clock divider values and resulting clock frequency

Divider	Resulting Frequency
2	125MHz
4	62.5MHz
6	41.6MHz
8	31.25MHz
10	25MHz
20	12.5MHz
50	5MHz
100	2.5MHz
200	1.25MHz

Data Control

The data control section is used to set the magnitude of the selected data pattern. Furthermore, there is a control bit that determines whether to use externally sourced data (from the GUI), or internally sourced data within the FPGA and sourced from a Linear Feedback Shift Register (LFSR)). An example setting is shown in Figure D.9.

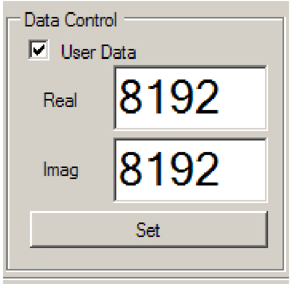

Figure D.9: Data amplitude area showing real and imaginary amplitude adjustment

Acquisition Control

To aid data pattern generation it is possible to select source data patterns based on a decimal number. Furthermore, any available value can be directly typed in where only a specific value is required. An illustration is provided in Figure D.10.

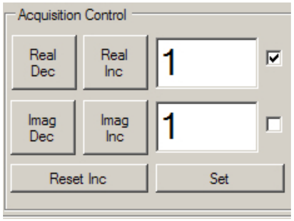

Figure D.10: Acquisition control area showing data sequence controlled by a counter (left) or an input integer value (right)

User Data

Each of the real and imaginary data ports can be set to positive, negative or zero (off state). Data with either a positive or negative value is multiplied by the corresponding Real or Imaginary value (as defined in *Data Control above*). The value of these ports is used as the source data for SEFDM signal generation. Figure D.11 illustrates a configuration that would result in a signal with 8 sub-carriers all with positive modulation. Figure D.12 illustrates the same configuration but with both positive and negative modulation on 8 sub-carriers. The modulation is still applied only to the real ports. As a final example, Figure D.13 shows data applied to both the real and imaginary ports. Any combination of positive, negative, real and imaginary is

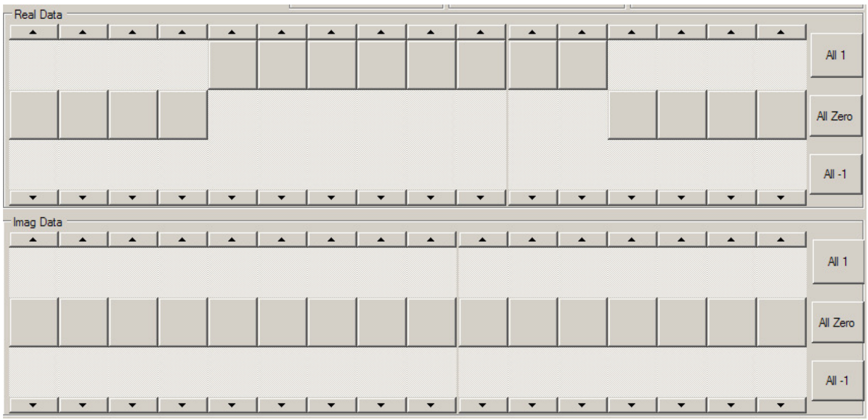

Figure D.11: Example use case of real-only data with positive-only amplitude

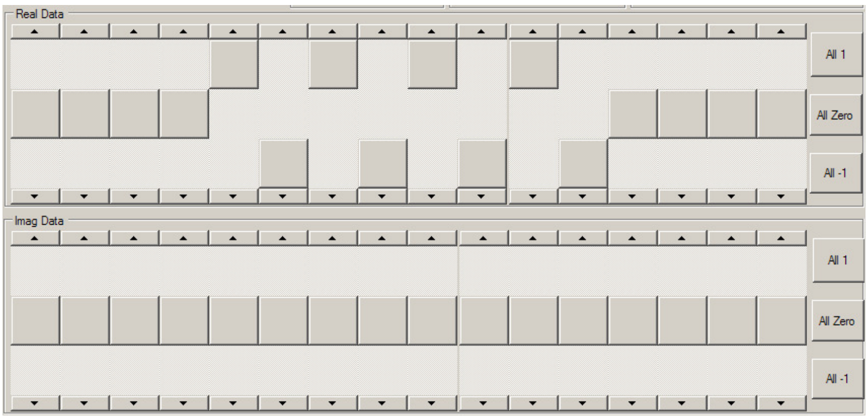

Figure D.12: Example use case of real-only data with positive and negative amplitude

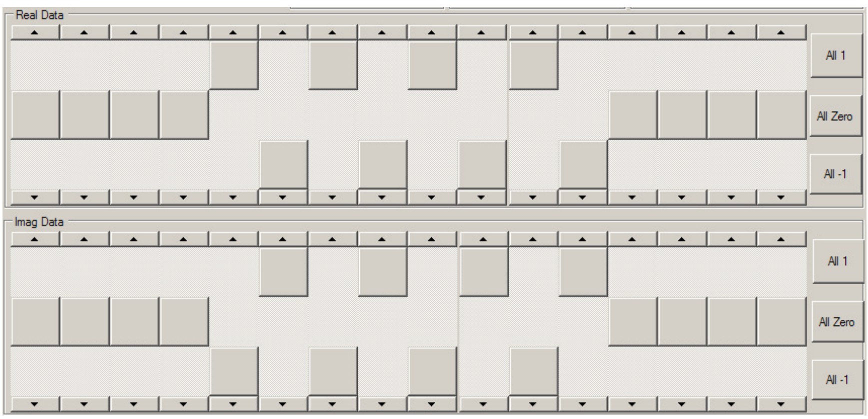

Figure D.13: Example use case of real and imaginary data combined with positive and negative amplitude

supported. Importantly, all of the real and all of the imaginary values will be of the same magnitude which is determined by the value set in the *Data Control* section.

Appendix E

Source code

This appendix details the code structure of the code provided in the companion Compact Disc. There are three types of code detailed; namely Very high speed integrated circuit Hardware Description Language (VHDL) (for the Field Programmable Gate Array (FPGA) Spectrally Efficient Frequency Division Multiplexing (SEFDM) implementation) in Section E.1, C++ (for the Graphical User Interface (GUI)) in Section E.3 and Gerber (for the Printed Circuit Board (PCB)) in Section E.2.

E.1 VHDL Code

The VHDL code hierarchy stems from a master file, *Top_Level.vhd*. This file contains instances of all the files listed below and in addition, defines physical pin names (internal to the FPGA) and manages resets and clocks. The code is found in the *VHDL* folder on the companion CS. To produce a working FPGA file the .xco files need regenerating (using vendor tools as appropriate) and a new project with all the detailed files included. Table E.1 defines the name of each file in the folder and a summary of its purpose.

Table E.1: VHDL Code heirarchy and Description

File Name	**Level**	**Description**
top_level_package	1	Definitions file
tricmac_block	1	Ethernet wrapper
rgmii_v2.0_if.vhd	2	Ethernet interface
emac0_fcs_blk_rgmii.vhd	2	Ethernet general
trimac.vhd	2	Ethernet core
enet.vhd	1	Code from Chapter 4
mac_rx_fifo.vhd	2	Ethernet RX buffer
mac_tx_fifo.vhd	2	Ethernet RX buffer
data_gen.vhd	1	Random data generator code
CPX_SEFDM_REORDER_INDEX.vhd	1	Code from Chapter 5
CPX_SEFDM_IFFT.vhd	1	Code from Chapter 5
xfft_fixedpoint_single.xco	2	Inverse Fast Fourier Transform (IFFT) code
mp_div.vhd	2	Bit-shifting divider
CPX_SEFDM_ROTATE.vhd	1	Code from Chapter 5
dft.xco	2	Inverse Discrete Fourier Transform (IDFT) code
cpx_m_coregen.vhd	2	Complex multiplier
vot_vec_ram.vhd	2	Memory to store rotation vectors
data_buffer.xco	1	Digital to Analogue Converter (DAC) interface buffer
sine_rom_small.vhd	1	Memory to store sinusoid for debug
top_level_ucf.ucf	1	Pin and timing definition file

E.2 Gerber Code

The code that describes the PCB, detailed in 2 is supplied in the *gerber* folder on the companion CD. Each of the supplied files is described in Table E.2 and can be displayed using a gerber viewing tool.

Table E.2: Gerber code description

File Name	**Description**
top.gbr	Top layer track layout
bottom.gbr	Bottom layer track layout
mid1.gbr	Internal signal layer
mid2.gbr	Internal signal layer
mid3.gbr	Internal signal layer
mid4.gbr	Internal signal layer
mid5.gbr	Internal signal layer
mid6.gbr	Internal signal layer
mid7.gbr	Internal signal layer
mid8.gbr	Internal signal layer
mid9.gbr	Internal signal layer
mid10.gbr	Internal signal layer
mid11.gbr	Internal signal layer
mid12.gbr	Internal signal layer
mid13.gbr	Internal signal layer
mid14.gbr	Internal signal layer
plane1.gbr	Ground plane layer
plane2.gbr	Ground plane layer
plane3.gbr	Ground plane layer
plane4.gbr	Ground plane layer
plane5.gbr	Ground plane layer
plane6.gbr	Ground plane layer
plane7.gbr	Ground plane layer
plane8.gbr	Ground plane layer
drill.gbr	Drill file for via hole placement
topoverlay.gbr	Top silk-screen
topsolder.gbr	Top Solder position
toppaste.gbr	Top Solder Paste Position
bottomoverlay.gbr	Bottom silk-screen
bottomsolder.gbr	Bottom Solder position
bottompaste.gbr	Bottom Solder Paste Position

E.3 C++ Code

The code for the GUI detailed in Appendix D is provided in the *C++* folder on the companion CD. This code forms part of a Microsoft Visual C++ 2010 project (named *SEFDM_Control*). To ensure that the code can be reused and recompiled using future versions all files (both written by the author and provided by the tools) are included. Table E.3 details the files provided with a summary of function.

Table E.3: C++ Code heirarchy and Description

File Name	**Level**	**Description**
Form1.h	1	Graphics definition
Form1.resx	2	Graphic description
resource.h	1	Windows required file
stdafx.h	1	Windows required file
StringConverter.h	1	Windows required for string class
app.ico	1	Executable Icon
app.rc	1	Executable resource
AssemblyInfo.cpp	1	Windows generated code hierarchy
SEFDM_Control.cpp	1	Authors code
stdafx.cpp	1	Windows required file

Appendix F

Cordic Details

The COordinate Rotation DIgital Computer (CORDIC) is a method of generating complex vectors which represent sine or cosine components. Formally, a vector (X, Y) is rotated through an angle θ which results in $(X^{'}, Y^{'})$. As opposed to direct implementations to compute sine and cosine which require multiplication, a CORDIC requires only bit-shift, add and subtraction operations. Hence, its application is attractive in logic devices. A CORDIC operation is performed by limiting rotation angles to $atan(2^{-i})$, where $0 \leq i < n$ and are hereby termed *micro rotations*. Each micro rotation can be expressed for each i^{th} value as:

$$X_{i+1} = X_i - \alpha_i Y_i 2^{-i} \tag{F.1}$$

$$Y_{i+1} = Y_i - \alpha_i X_i 2^{-i} \tag{F.2}$$

$$\theta_{i+1} = \theta_i - \alpha_i atan(2^{-i}) \tag{F.3}$$

where $\alpha_i = \pm 1$. In terms of a vector rotation, defined as n series of micro rotations, the resulting vectors, $(X^{'}, Y^{'})$ are found by:

$$X^{'} = \prod_{i=1}^{n} cos(tan^{-1}(2^{-i}))(X_i - \alpha Y_i 2^{-i}) \tag{F.4}$$

$$Y^{'} = \prod_{i=1}^{n} cos(tan^{-1}(2^{-i}))(Y_i - \alpha X_i 2^{-i}) \tag{F.5}$$

$$\theta^{'} = \sum_{i=1}^{n} \theta - (\alpha_i tan^{-1}(2^{-1})) \tag{F.6}$$

By selecting α_i such that the resulting $\theta^{'} - > 0$, CORDIC enables the rotation of Cartesian vectors by a required phase angle, θ:

$$X' = Z_n(cos(\theta)X - sin(\theta)Y) \tag{F.7}$$

$$Y' = Z_n(cos(\theta)Y + sin(\theta)X) \tag{F.8}$$

$$\theta' = 0 \tag{F.9}$$

where

$$Z_n = \frac{1}{\prod_{i=0}^{n} cos^{-1}(tan^{-1}(2^{-i}))} \tag{F.10}$$

For example, given an input vector $(0.707, 0.25)$ and a required phase shift of $\pi/2$, the resulting vector would be $(0.25, -0.707)$. The translation of Cartesian vectors to the equivalent polar notation is performed by rotation of the Cartesian vectors, (X, Y), around a unit circle until $Y = 0$; formally expressed as:

$$X' = Z_n\sqrt{(X^2 + Y^2)} \tag{F.11}$$

$$zY' = 0 \tag{F.12}$$

$$\theta = tan^{-1}(\frac{Y}{X}) \tag{F.13}$$

Using the same input vectors as previously, $(0.707, 0.25)$, the conversion to polar form results in $r = 0.75$ and $\theta = 0.336$.

References

[1] Xilinx. Advantages of the Virtex-5 FPGA 6-Input LUT Architecture. Technical report, 2007.

[2] Alan Mishchenko, Satrajit Chatterjee, and Robert K. Brayton. Improvements to Technology Mapping for LUT-Based FPGAs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 26(2):240–253, February 2007.

[3] Cristian Grozea and Zorana Bankovic. FPGA vs. multi-core CPUs vs. GPUs: hands-on experience with a sorting application. *Facing the multicore-challenge*, pages 1–12, 2011.

[4] Jing Ma and Xinming Huang. A System-on-Programmable Chip Approach for MIMO Sphere Decoder. *13th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'05)*, pages 317–318, 2005.

[5] Khaled Sobaihi, Akram Hammoudeh, and Scammell David. FPGA Implementation of OFDM Transceiver for a 60GHz Wireless Mobile Radio System. In *IEEE International Conference on Reconfigurable Computing and FPGAs, (ReConFig'2010)*, pages 185–189, 2010.

[6] J. Cumplido and R. Garcia. On the Design of an FPGA-Based OFDM Modulator for IEEE 802.16-2004. In *IEEE International Conference on Reconfigurable Computing and FPGAs, (ReConFig'2005)*, pages 22–25, 2005.

[7] M. Santhi, Sowjanya Tungala, C. Balakrishna, and G. Lakshminarayanan. Asynchronous pipelined MB-OFDM UWB transceiver on FPGA. In *IEEE Region 10 TENCON Conference*, pages 1–5. IEEE, November 2010.

[8] Sherif Kishk, Ahmed Mansour, and Mohy Eldin. Implementation of an OFDM System Using FPGA. *26th National Radio Science Conference.*, 2009.

[9] Jinsong Xu, Xiaochun Lu, Haitao Wu, Yujing Bian, Decai Zou, Xiaolong Zou, and Chaogang Wang. Implementation of MB-OFDM Transmitter Baseband Based on FPGA. *4th IEEE International Conference on Circuits and Systems for Communications*, (2006):50–54, May 2008.

[10] Aifeng Ren, Ming Luo, and Fangming Hu. FPGA implementation of an OFDM modem. *IET International Communication Conference on Wireless Mobile and Computing (CCWMC)*, pages 761–764, 2009.

[11] Xiaoxin Cui and Dunshan Yu. Digital OFDM transmitter architecture and FPGA design. *IEEE 8th International Conference on ASIC*, pages 477–480, October 2009.

[12] K. Harikrishna, T. Rama Rao, and Valadimir A. Labay. An FFT Approach for Efficient OFDM Communication Systems. *International Conference on Signal Acquisition and Processing*, (2):117–119, February 2010.

[13] J. Veilleux, P. Fortier, and S. Roy. An FPGA Implementation of an OFDM Adaptive-Modulation System. *3rd International NEWCAS Conference*, pages 60–63, 2005.

[14] Fangming Liu and Xiaozhong Pan. Research on Implementation of FFT Based on FPGA. *International Conference on Computer Applications and System Modeling (ICCASM)*, (Iccasm):152–155, 2010.

[15] Xinming Huang, Cao Liang, and Jing Ma. System Architecture and Implementation of MIMO Sphere Decoders on FPGA. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 16(2):188–197, February 2008.

[16] Andreas Burg. *VLSI Circuits for MIMO Communication Systems.* PhD thesis, Swiss Federal Institute of Technology, 2006.

[17] Heejung Yu, Kyonghee Song, Kwhanghyun Ryu, Yunjoo Kim, Seungwook Min, and Sok-kyu Lee. Design and FPGA implementation of MIMO-OFDM based WLAN Systems. *63rd IEEE Vehicular Technology Conference (VTC)*, pages 1333–1338, 2006.

[18] Xiang Wu and J.S. Thompson. FPGA implementation of an efficient high-throughput sphere decoder for MIMO systems based on the smallest singular value threshold. In

IEEE Conference on Adaptive Hardware and Systems (AHS), number 1, pages 340–345, 2010.

[19] I. Az, S. Sahin, and M.A. Cavuslu. Implementation of Fast Fourier and Inverse Fast Fourier Transforms in FPGA. In *IEEE 15th Signal Processing and Communications Applications*, pages 1–4. IEEE, 2007.

[20] J. Viejo, A. Millan, M.J. Bellido, E. Ostua, and A. Munoz. Implementation of a FFT/IFFT Module on FPGA: Comparison of Methodologies. In *4th Southern Conference on Programmable Logic*, volume 2, pages 7–11. IEEE, 2008.

[21] F. Rusek and J.B. Anderson. The two dimensional Mazo limit. *Proceedings of International Symposium on Information Theory (ISIT)*, 0(1):970–974, 2005.

[22] Safa Isam and Izzat Darwazeh. Simple DSP-IDFT Techniques for Generating Spectrally Efficient FDM System Signals. *7th International Symposium on Communication Systems Networks and Digital Signal Processing (CSNDSP)*, 2010.

[23] M. Hamamura and S. Tachikawa. Bandwidth efficiency improvement for multi-carrier systems. *IEEE 15th International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, 1:48–52, 2004.

[24] Wang Jian, Yang Xun, Zhang Xi-lin, and Daoben Li. The Prefix Design and Performance Analysis of DFT-based Overlapped Frequency Division Multiplexing (OvFDM-DFT) System. *3rd International Workshop on Signal Design and Its Applications in Communications*, (Ic):361–364, September 2007.

[25] A. Sghaier, S. Areibi, and B. Dony. A Pipelined Implementation of OFDM Transmission on Reconfigurable Platforms. In *Canadian Conference on Electrical and Computer Engineering (CCECE)*, pages 000801–000804. IEEE, 2008.

[26] Johanna Ketonen, Markku Juntti, and Joseph R. Cavallaro. PerformanceComplexity Comparison of Receivers for a LTE MIMOOFDM System. *IEEE Transactions on Signal Processing*, 58(6):3360–3372, June 2010.

[27] M.S. Naghmash, M.F. Ain, and C.Y. Hui. FPGA Implementation of Software Defined Radio Model based 16QAM. *European Journal of Scientific Research (EJSR)*, 10(2):301–310, 2009.

[28] Y.S. Kwon and C.M. Kyung. ATOMi: an algorithm for circuit partitioning into multiple FPGAs using time-multiplexed, off-chip, multicasting interconnection architecture. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 13(7):861–864, 2005.

[29] J.L. Beuchat and J.M. Muller. Modulo M multiplication-addition: algorithms and FPGA implementation. *Electronics Letters*, 40(11):654–655, 2004.

[30] N. Matsumoto, Koichi Ichige, and H. Arai. Fixed-point digital processing of recursive least-square algorithm toward FPGA implementation of MMSE adaptive array antenna. In *7th IEEE International Symposium on Signal Processing and Its Applications.*, volume 2, pages 615–617, 2003.

[31] D. Hammerstrom. Platform performance comparison of PALM network on Pentium 4 and FPGA. *Proceedings of the International Joint Conference on Neural Networks*, pages 995–1000, 2003.

[32] DB Thomas and Lee Howes. A comparison of CPUs, GPUs, FPGAs, and massively parallel processor arrays for random number generation. *Proceedings of the ACM/SIGDA international symposium on Field programmable gate arrays*, 2009.

[33] L.G. Barbero and J.S. Thompson. FPGA design considerations in the implementation of a fixed-throughput sphere decoder for MIMO systems. In *International Conference on Field Programmable Logic and Applications*, pages 1–6, 2007.

[34] M. S. Khairy, M. M. Abdallah, and S. E.-D. Habib. Efficient FPGA Implementation of MIMO Decoder for Mobile WiMAX System. *IEEE International Conference on Communications*, pages 1–5, June 2009.

[35] Maxim Semiconductor - www.maxim-ic.com.

[36] Analog Devices - www.analog.com.

[37] Texas Instruments - www.ti.com.

[38] NXP Semiconductor - www.nxp.com.

[39] Linear Technology - www.linear.com.

[40] Xilinx - www.xilinx.com.

[41] Altera - www.altera.com.

[42] Lattice Semiconductor - www.latticesemi.com.

[43] Actel - www.actel.com.

[44] Micron Semiconductor - www.micron.com.

[45] IDT - www.IDT.com.

[46] GSI Technology - www.GSItech.com.

[47] Cypress Semiconductor - www.cypress.com.

[48] Xilinx. Virtex-5 FPGA RocketIO tranciever user guide. Technical report, 2007.

[49] Xilinx. Embedded Tri-Mode Ethernet MAC. Technical report, 2010.

[50] Xilinx. Virtex-5 FPGA Packaging and Pinout Specification. Technical report, 2010.

[51] Xilinx. Xilinx FPGA configuration and user guide. Technical report, 2011.

[52] Walt Kester. Evaluating High Speed DAC Performance. Technical report, 2008.

[53] Analog Devices AD9779 Datasheet. Technical report.

[54] Analog Devices AD5370. Technical report, 2006.

[55] Analog Devices AD8003 Datasheet. Technical report, 2008.

[56] Cosoroaba (Xilinx) Adrian. Xilinx MIG user guide. Technical report, 2007.

[57] Virtex-5 Family Overview. Technical report, 2007.

[58] National Semiconductor DP83865 Datasheet. Technical report, 2004.

[59] Micron LPDDR RAM MT46H32M32. Technical report, 2008.

[60] Xilinx. Xilinx XPower Estimator User Guide. Technical report, 2012.

[61] John Rice, Dirk Gehrke, and Mike Segal. Understanding Noise-Spreading Techniques and their Effects in Switch-Mode Power Applications. Technical report.

[62] Rick Downs. Partitioning and layout of a mixed signal PCB. Technical report.

[63] Henry Ott. Partitioning and Layout of a mixed signal PCB. *Printed Circult Design*, 2001.

[64] Rick Downs. Configure analogue, Digital grounds. *EETimes-India*, pages 1–2, 2009.

[65] Bruce (TI) Carter. TI - How (Not) to Decouple High-Speed Operational Amplifiers. Technical Report 1, 2001.

[66] V. Ricchiuti. Power-supply decoupling on fully populated high-speed digital PCBs. *IEEE Transactions on Electromagnetic Compatibility*, (4):671 – 676, 2001.

[67] Amphenol - www.amphenol.com.

[68] Xilinx. Virtex-5 PCB Designer Guide. Technical report, 2007.

[69] Micron. LPDDR2 RAM PCB Design guide. Technical report, 2008.

[70] Leo (National Semiconductor) Chang and Patrick (National Semiconductor) O Farrell. DP83865 PCB Design Guide. Technical report, 2008.

[71] Micrel Semiconductor - www.micrel.com.

[72] Xilinx. Xilinx Virtex-5 FPGA User Guide. Technical report, 2012.

[73] Marcus Perrett and Izzat Darwazeh. Flexible Hardware Architecture of SEFDM Transmitters with Real-Time Non-Orthogonal Adjustment. In *18th IEEE International Conference on Telecommunications*, 2011.

[74] Paul N Whatmough, Marcus Perrett, Safa Isam, and Izzat Darwazeh. VLSI Architecture for a Reconfigurable Spectrally Efficient FDM Baseband Transmitter. *IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–4, 2011.

[75] Lecroy. Application Note - Enhanced Resolution. Technical report.

[76] Marcus Perrett. Implementation of a M-Sequence Pseudo Random Binary Sequence audio measurement system based on the Hadamard transform. *London Communication Symposium (LCS)*, (6):1–4, 2010.

[77] Syed Qasim and Shuja Abbasi. Single chip FPGA based realization of arbitrary waveform generator using rademacher and walsh functions. *International Conference on Emerging Technologies*, (November):205–210, 2006.

[78] Ray Andraka. A survey of CORDIC algorithms for FPGA based computers. *Proceedings of the 6th international symposium on Field programmable gate arrays*, pages 191–200, 1998.

[79] Maher Jridi and Ayman Alfalou. Direct Digital Frequency Synthesizer with CORDIC Algorithm and Taylor Series Approximation for Digital Receivers. *European Journal of Scientific Research*, 30(4):542–553, 2009.

[80] Lecroy. Computation of Effective Number of Bits, Signal to Noise Ratio, & Signal to Noise & Distortion Ratio Using FFT. Technical report, 2011.

[81] Mentor Graphics IP Cores - www.mentor.com.

[82] Hitech Global - www.hitechglobal.com.

[83] Intilop - www.intilop.com.

[84] Liu Tian-Hua, Zhu Hong-Feng, Liu Jun, Zhou Chuan-Sheng, and Chang Gui-Ran. Design and Implementation of ARP Based on VHDL. *6th International Conference on ITS Telecommunications*, 0:20–25, 2006.

[85] F.L. Herrmann, Guilherme Perin, J.P.J. de Freitas, Rafael Bertagnolli, and J.B. dos Santos Martins. A Gigabit UDP/IP network stack in FPGA. In *16th IEEE International Conference on Electronics, Circuits and Systems*, pages 836–839. IEEE, December 2009.

[86] A. Lofgren, L. Lodesten, S. Sjoholm, and H. Hansson. An analysis of FPGA-based UDP/IP stack parallelism for embedded Ethernet connectivity. *Norchip*, pages 94–97, 2005.

[87] Wang Renbo and W. Xiong. Reduced TCP/IP Protocol Implement in VHDL. In *International Workshop on Intelligent Systems and Applications*, number C, pages 1–5. IEEE, 2009.

[88] Yatin Hoskote, Bradley A Bloechel, Gregory E Dermer, Vasantha Erraguntla, David Finan, Jason Howard, Dan Klowden, Siva G Narendra, Greg Ruhl, James W Tschanz, Sriram Vangal, Venkat Veeramachaneni, Howard Wilson, Jianping Xu, and Nitin Borkar. A TCP Offload Accelerator for 10 Gb / s Ethernet in 90-nm CMOS. *IEEE Journal of solid-state circuits*, 38(11):1866–1875, 2003.

[89] Krishna Kant. TCP offload performance for front-end servers. *IEEE Global Telecommunications Conference*, pages 3242–3247, 2003.

[90] Jian Wang, Hong Wang, and Zhi-jia Yang. An FPGA based slave communication controller for Industrial Ethernet. *9th International Conference on Solid-State and Integrated-Circuit Technology*, pages 2062–2065, October 2008.

[91] Marcus Perrett and Izzat Darwazeh. FPGA Implementation of Quad Output Generator for Spectrally Efficient Wireless FDM Evaluation. *7th International Symposium of Broadband Communications (ISBC)*, 2010.

[92] Wireshark - www.wireshark.com.

[93] Russell Tessier and Heather Giza. Balancing logic utilization and area efficiency in FPGAs. *IEEE 10th International Workshop on Field-Programmable Logic and Applications (FPL)*, pages 535–544, 2000.

[94] Michael Gschwind and Valentina Salapura. A VHDL Design Methodology for FPGAs. *Proceedings of the 5th International Workshop on Field-Programmable Logic and Applications (FPL)*, pages 208 – 217, 1995.

[95] Steve Golson. State Machine Design Techniques for Verilog and VHDL. Technical Report September 1994, 1994.

[96] Ryan Grammenos and Izzat Darwazeh. SC-FDMA and OFDMA : The two competing technologies for LTE. *4th International Symposium on Broadband Communications (ISBC)*, 2010.

[97] Neetu Sood, A.K. Sharma, and M. Uddin. BER performance of OFDM-BPSK and-QPSK over Nakagami-m fading channels. In *2nd IEEE International Conference on Advance Computing Conference (IACC)*, number 1, pages 88–90, 2010.

[98] Safa Isam and Izzat Darwazeh. IDFT Based Transmitters for Spectrally Efficient FDM System,. *London Communication Symposium (LCS)*, 2010.

[99] Ioannis Kanaras, Arsenia Chorti, M.R.D. Rodrigues, and Izzat Darwazeh. Investigation of a Semidefinite Programming detection for a spectrally efficient FDM system. *20th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, pages 2827–2832, September 2009.

[100] Ioannis Kanaras, Arsenia Chorti, M.R.D. Rodrigues, and Izzat Darwazeh. Spectrally Efficient FDM Signals: Bandwidth Gain at the Expense of Receiver Complexity. *IEEE International Conference on Communications*, 1:1–6, June 2009.

[101] Ioannis Kanaras, Arsenia Chorti, M.R.D. Rodrigues, and Izzat Darwazeh. A new quasi-optimal detection algorithm for a non orthogonal Spectrally Efficient FDM. *9th International Symposium on Communications and Information Technology*, pages 460–465, September 2009.

[102] M. Joham, L.G. Barbero, T. Lang, W. Utschick, J.S. Thompson, and T. Ratnarajah. FPGA implementation of MMSE metric based efficient near-ML detection. In *IEEE International ITG Workshop on Smart Antennas (WSA)*, pages 139–146, 2008.

[103] J.E. Mazo. Faster-than-Nyquist signaling. *Bell Syst. Tech. J*, 54(8):1451–1462, 1975.

[104] Deepak Dasalukunte, Fredrik Rusek, Viktor Owall, Karthik Ananthanarayanan, and Mur Kandasamy. Hardware Implementation of Mapper for Faster-Than-Nyquist Signaling Transmitter. *IEEE Norchip Conference*, pages 1–5, November 2009.

[105] B. Le Floch, M. Alard, and C. Berrou. Coded orthogonal frequency division multiplex [TV broadcasting]. *Proceedings of the IEEE*, 83(6):982–996, June 1995.

[106] J.P. Javaudin and D. Lacroix. Technical description of the OFDM/IOTA modulation. Technical report, France Telecom R&D, Tech. Rep. R1-03-168, 2002.

[107] D. Lacroix, N. Goudard, and M. Alard. OFDM with guard interval versus OFDM/offsetQAM for high data rate UMTS downlink transmission. In *Proceedings of IEEE 54th Vehicular Technology Conference (VTC)*, pages 2682–2686, 2001.

[108] Deepak Dasalukunte, Fredrik Rusek, John B. Anderson, and Viktor Owall. Transmitter architecture for faster-than-Nyquist signaling systems. *IEEE International Symposium on Circuits and Systems*, (1):1028–1031, May 2009.

[109] John B. Anderson and Fredrik Rusek. Improving OFDM : Multistream Faster-than-Nyquist Signaling. *4th International Symposium on Turbo Codes & Related Topics; 6th International ITG Conference on Source and Channel Coding*, 2006.

[110] Deepak Dasalukunte, Fredrik Rusek, and Viktor Owall. An Iterative Decoder for Multicarrier Faster-Than-Nyquist Signaling Systems. In *IEEE International Conference on Communications (ICC)*, pages 1–5. IEEE, 2010.

[111] Deepak Dasalukunte, Fredrik Rusek, and Viktor Öwall. Multicarrier Faster-Than-Nyquist Transceivers Hardware Architecture and Performance Analysis. *IEEE Transactions on Circuits and Systems I: Regular Papers*, pages 1–12, 2010.

[112] Lionel Cordesses. Direct Digital Synthesis: A Tool For Periodic Wave Generation (Part 2). *IEEE Signal Processing Magazine*, 50(12):110–117, December 2004.

[113] Paul N Whatmough, Marcus Perrett, Safa Isam, and Izzat Darwazeh. VLSI Architecture for a Reconfigurable Spectrally Efficient FDM Baseband Transmitter. *IEEE Transactions of Circuits and Systems*, 59(5), 2012.

[114] G. Marcus, P. Hinojosa, A. Avila, and J. Nolazco-Flores. A fully synthesizable single-precision, floating-point adder/substractor and multiplier in VHDL for general and educational use. *Proceedings of the 5th IEEE International Caracas Conference on Devices, Circuits and Systems*, pages 319–323, 2004.

[115] C.C. Doss and R.L. Riley. FPGA-Based Implementation of a Robust IEEE-754 Exponential Unit. *12th Annual IEEE Symposium on Field-Programmable Custom Computing Machines*, pages 229–238, 2004.

[116] Carlos Minchola and Gustavo Sutter. A FPGA IEEE-754-2008 Decimal64 Floating-Point Multiplier. *International Conference on Reconfigurable Computing and FPGAs*, pages 59–64, December 2009.

[117] M.R.D. Rodrigues and Izzat Darwazeh. A spectrally efficient frequency division multiplexing based communications system. *In Proceedings of 2nd International Symposium on Broadband Communications (ISBC)*, pages 48 – 49, 2006.

[118] Safa Isam and Izzat Darwazeh. Peak to Average Power Ratio Reduction in Spectrally Efficient FDM Systems. *18th IEEE International Conference on Telecommunications*, (2), 2011.

[119] Ioannis Kanaras, Arsenia Chorti, and M.R.D. Rodrigues. A Fast Constrained Sphere Decoder for Ill Conditioned Communication Systems. *IEEE Communications Letters*, 14(11):999–1001, 2010.

[120] Safa Isam, Ioannis Kanaras, and Izzat Darwazeh. A Truncated SVD Approach for Fixed Complexity Spectrally Efficient FDM Receivers. *IEEE Wireless Communications & Networking Conference (WCNC)*, 2011.

[121] Safa Isam and Izzat Darwazeh. Design and Performance Assessment of Fixed Complexity Spectrally Efficient FDM Receivers. *73rd IEEE Vehicular Technology Conference (VTC)*, 2011.

[122] Ryan C Grammenos, Safa Isam, and I. Darwazeh. FPGA Design of a Truncated SVD Based Receiver for the detection of SEFDM Signals. In *IEEE Int. Symp. Personal, Indoor and Mobile Radio Communication*, 2011.

[123] M.R.D. Rodrigues and Izzat Darwazeh. Fast OFDM: A Proposal for Doubling the Data Rate of OFDM Schemes. *Proceedings of the International Conference on Telecommunications (ICT)*, 3:484–487, 2002.

[124] E. Agrell, T. Eriksson, A. Vardy, and K. Zeger. Closest point search in lattices. *IEEE Transactions on Information Theory*, 48(8):2201–2214, 2002.

[125] Ryan C Grammenos and I. Darwazeh. Performance Trade-Offs for the Detection of Non-Orthogonal FDM signals with a Practical Receiver Complexity. In *Submitted in GLOBECOM*, 2012.

[126] Ryan C Grammenos and I. Darwazeh. Hardware Implementation of a Practical Complexity Spectrally Efficient FDM Reconfigurable Receiver. In *IEEE Int. Symp. Personal, Indoor and Mobile Radio Communication (PIMRC)*, 2012.